Interactive Query Languages for External Data Bases

F.H. Lochovsky

D.C. Tsichritzis

March, 1981

Computer Systems Research Group
University of Toronto
Toronto, M5S 1A1
Canada

For: Department of Communications, Ottawa, Canada
Under: Department of Supply & Services Contract Serial No. 0SU80-00124
Period: July 1, 1980 - March 31, 1981

Copies of

TELIDON BEHAVIOURAL RESEARCH 5
INTERACTIVE QUERY LANGUAGES FOR EXTERNAL DATA BASES

are available
free of charge from:

Information Services
Department of Communications
300 Slater Street
Ottawa, Ontario
K1A 0C8

Report Number DOC-TBR-DBRE-81-5-E
(Printed in March, 1982)

ABSTRACT

The amount of information that is available in computerized form is growing steadily. Much of this information is of an informational or reference nature and could potentially be made available to the general public. Until recently, there was no convenient medium by which the public could gain access to this information. However, two developments — the growth of communication networks and the advent of videotex systems — provide the potential for allowing the public convenient access to this information. In this report we explore the nature of the mechanism by which people in their homes using a videotex information service might access data bases external to the videotex system itself.

We first consider the process of accessing a data base (querying) in terms of the user interaction characteristics of the process. We divide the querying process into three parts: request, reply and dynamics. For all three parts, certain characteristics are identified that describe the parameters of the user interaction. For requests, the parameters are number of keystrokes required, type and number of commands available, degree of freedom in request formulation, selectivity of requests, uniformity of requests with respect to data bases accessed and ability to customize the request formulation for specific applications. For replies, the parameters are the form in which the reply is presented to the user, whether the reply can be presented via more than one medium, whether the reply can be customized, the degree of dynamic control over the reply and the possibility of saving the reply and using it as input at some later time. For the dynamics of interaction, the parameters relate to the bandwidth between the user and the system, the degree of gamesmanship involved in the interaction, number of protocols available for interaction, the responsiveness of the system to a user request and the degree of control experienced by the user in an interaction. For each of these parameters, a desirable "value" for the parameter is identified.

We then identify seven different query language types: keyword, by example, natural language, menu, graphic, multi-media and computer games. Each query language type is described and evaluated in terms of the interaction parameters.

From this evaluation of query languages, it is obvious that no one type of query language has all of the desirable characteristics for a query language. We therefore propose some requirements that a query language for external data bases should have. With these requirements in mind we then propose a design for a query language for accessing external data bases.

For describing the query language, a paradigm of navigating a personal spaceship through space using detailed maps for guidance is used. Basically the user is able to request maps of the information space available to him and to choose his destination from these maps. The maps are presented in the form of templates which show the structure and/or contents of the external data bases. Most of the interaction is accomplished by pointing at fields of the templates and either filling in values or requesting a display of values. The nature of the interaction allows formatted data, nonformatted text data, image data and voice data to be manipulated in a uniform manner.

CONTENTS

1 INTRODUCTION 5

2 USER INTERACTION PARAMETERS 6
- 2.1 Request 6
- 2.2 Reply 8
- 2.3 Dynamics 9
- 2.4 Summary 10

3 QUERY LANGUAGE EVALUATION 12
- 3.1 Keyword 12
- 3.2 By Example 13
- 3.3 Natural Language 14
- 3.4 Menu 15
- 3.5 Graphic 16
- 3.6 Multi-media 16
- 3.7 Computer Games 17

4 QUERYING REQUIREMENTS 19

5 PARADIGM 21

6 DESIGN OVERVIEW 23
- 6.1 Choosing An External Data Base 23
- 6.2 Exploring the Data 26
- 6.3 Predefined Routes 27

7 DESIGN CONSIDERATIONS 28
- 7.1 Pointing 28
- 7.2 Portal Positioning 29
- 7.3 Control Functions 30
- 7.4 Specification Modes 31
- 7.5 Data Types 32

8 CONCLUDING REMARKS 34

BIBLIOGRAPHY 36

1 INTRODUCTION

Recent years have seen a substantial growth in the number of on-line, computer-based information sources (data bases). Many of these on-line data bases provide information to the public, e.g., library systems, airline systems and government information services. However, the public generally does not have convenient and direct access to these data bases. Instead, some intermediary queries the data base and supplies the response, or the information is available in a non-selective manner in fixed locations.

Another development in recent years has been the installation and growth of communication networks. These communication networks allow data bases to be queried remotely at reasonable cost. This in turn raises the possibility of permitting access, by the public, to on-line "consumer information" data bases. An important question in this context is from where and how is this access to be permitted.

A third development in recent years has been the emergence of videotex systems. *Videotex systems* are interactive, visual communication systems intended, in part, to permit public access to external data bases. (By an *external* data base we mean one whose contents and format is not under the control of the information delivery (videotex) system.) Telidon is an example of such a system [Bown *et al.*, 1978]. The interaction with the system is via a suitably modified television receiver which acts as the terminal display. Input is accomplished by a keypad or keyboard which allows certain data to be selected for display.

Given that the public will eventually have access to videotex systems, several issues relating to user interaction with these systems arise [Bown *et al.*, 1979]. One of these issues concerns how the public will request information from the various external data bases that will be available to them. Currently, a number of different languages each with its own syntax and vocabulary exist for querying data bases.

It seems intuitively undesirable for a user of a videotex system to have to learn and remember a different query language for each external data base that will be accessed. To help ensure user acceptance of videotex information retrieval services, there should be one way of querying external data bases via videotex systems that is easy to learn and use by the public. Using this query language, people can access information in any external data base that is connected to the communication network.

In this report we identify and describe several categories of interactive query languages. We establish some user interaction criteria for evaluating these categories of query languages. We briefly discuss each category of query language with respect to these criteria. Finally, we propose a particular approach to a query language for external data bases.

2 USER INTERACTION PARAMETERS

The general subject of person-computer interaction has been the subject of much study [Martin, 1973; Gilb and Weinberg, 1977; Guedj *et al.*, 1980; Schneiderman, 1980; Mehlmann, 1981]. In the area of interactive query languages the studies are far fewer [Reisner, 1981]. Applying some of the results from general person-computer interaction and the studies on query languages, let us examine the process of querying a data base from the user's viewpoint. The querying process can be broken up into three parts: request, reply and dynamics.

Request deals with the formulation of a query to the system by the user. The user informs the system of some action that he wishes it to do for him. We are principally concerned here with requests for stored information. Usually some form of "language" that is mutually acceptable to the user and the system is used to specify a request. This language can take several forms, e.g., visual, verbal, pantomime, etc.

Reply deals with informing the user of the result of an action. The result can be the answer to a query or information about the status of a request. Presenting output to the user should be done in a way that is palatable to the user and easy for him to assimilate. Again, some form of "language" is required to communicate the output. Displayed output, either as tables, graphs, pictures, etc., seems to be the natural choice for videotex systems. However, the use of sound is also a possibility.

Finally, *dynamics* deals with the nature of the interaction between the user and the system. The request and reply processes must be accomplished via some communication medium between the user and the system, i.e., the "languages" must have some means of transport between the two "participants". For example, the interaction can be by typing, pointing, speaking, etc. The quality of this interaction, from the user's viewpoint, is a critically important aspect of the querying process. In videotex systems, we would like as much as possible that this interaction be easy and interesting for the user.

In most interactive query languages, these three issues are lumped together. This is a mistake since a language used for formulating requests has different user requirements than a language that is used for presenting replies. In addition, the dynamics of a language are independent of the static aspects of formulating requests or presenting replies. Thus requirements for the dynamics of a query language can be considered independent of request and reply requirements.

For each part of the querying process certain characteristics can be identified. These characteristics describe the parameters of user interaction. The "values" that these parameters have in a given query language determine whether or not the query language has "desirable" characteristics with regard to person-computer interface. In the rest of this section we will identify and define several characteristics for each part of the querying process. In addition, we will indicate what, in our opinion and that of the studies cited earlier, constitutes a "desirable" value for each characteristic. This list of "desirable" values will then constitute requirements for a "desirable" interactive query language.

2.1 Request

We identify six characteristics of formulating requests which are important for evaluating interactive query languages.

KEYSTROKES

Keystrokes are a quantification of the amount of user-supplied input required before the system fully understands the user's request. If requests are formulated by typing, then the number of characters that must be typed can be the measure used. If requests are formulated by pointing, then the number of things that must be pointed at can be used as a measure. It is desirable to *minimize* the number of keystrokes required to formulate a request. In this way requests can be formulated very quickly and easily and the possibility of some errors, e.g., typing errors, are reduced.

COMMANDS

Commands are the set of instructions a user has available to tell the system what action it should perform. For example, a user might want to tell the system to *retrieve* or *print* or *draw*. The number of commands available to a user should be *small* so that they can be remembered easily. The commands should also be *simple*. By this we mean that the syntax of a command and the number of things that need to be specified to use a command should be easy to remember and intuitive to the user.

FORMULATION

Formulation of a request concerns how difficult it is to specify a request that is incorrect either syntactically or semantically. A syntactically incorrect request is one that does not follow the "pattern" expected by the system. These usually consist of misspelled words or transposed or missing parts of a request. A semantically incorrect request is either one that cannot be answered by the system because it has no information about the request or one, for which the answer supplied, is not the reply the user expected. That is, the system's interpretation of the request differs from the user's interpretation of the request. It is desirable that a query language eliminate the possibility or make difficult the formulation of syntactically and/or semantically incorrect requests.

SELECTIVITY

Selectivity is the ability of the user to specify as precisely as possible that data which he wishes to retrieve. High selectivity of a query language implies that the user gets back only the data he wants. Low selectivity implies that much of the data that is retrieved is "noise" in that it is not relevant to the request. It is desirable that a query language have *high* selectivity. In this way the user is not distracted by irrelevant output. In addition, if the user is being charged according to the amount of data retrieved, then high selectivity may also minimize the cost of using the system.

UNIFORMITY

Uniformity concerns the independence of the query language from the type of data base it is accessing and the application for which the language is being used. Uniformity for data bases means, given several different types of data bases, some of which may be structured differently, can the query language be used to access all of these data bases. Uniformity for applications means, can the query language be used for any type of application or is the structure of a query application specific. For example, if a query language can be used to access stock, weather, news, grocery, etc. information, then it is uniform with respect to application. It is desirable that the structure of a query language be uniform with respect to both data bases and applications. In practice the latter is usually achieved

easily; the former is much more difficult.

CUSTOMIZING

Customizing of a query language deals with matching the form of a query language to that of a particular application. The intent is to make it easier to form requests for the application since the form of the query language helps the user remember the language. For example, certain applications use specific words for specific actions. These words could be used as keywords in a query language. In this way, the semantics of a request are conveyed more easily and precisely to the user. It is desirable that a query language employ as many application specific concepts as possible to increase user convenience in interacting with the system.

2.2 Reply

We identify five characteristics for representing replies which are important for evaluating interactive query languages.

PRESENTATION

Presentation is the form in which the reply is given to the user. The important consideration here is the complexity of the presentation. One measure of this complexity is how long it takes the user to grasp the content of the reply. If the reply is presented in dialogue form which the user must read on a display, then it can take quite some time to understand the information content of the reply. On the other hand, tabular or graphical presentations can usually be interpreted much faster. In addition, much more information can be presented in the same amount of space. It is desirable that the query language display the reply to a query in a form which is acceptable to the user and which is simple for the user to follow.

MULTI-MEDIA PRESENTATION

People communicate in more than one way. Therefore, as well as the form in which a reply is presented to the user, it is also desirable that the user be able to choose the medium by which he receives the reply. For example, the reply could be printed on paper, be displayed either as text, graphics or animation on a screen, or be in spoken form. In this way, the way in which the reply is presented could be matched to the moods and requirements of the user. He may want to leisurely browse through the reply, glance at it quickly, or listen to it while doing other things. The ability to direct the output to different devices and to different communication media may also be important for allowing handicapped people to use the system.

CUSTOMIZING

Query languages usually do not take cognizance of the audience for which the reply to a request is intended. There is usually a fixed way of presenting the reply to the user irregardless of application, type of display device, time of day or type of user. Customizing of replies, as for customizing of requests, means that the form of the reply is determined by the environmental factors of the interaction. For example, weather information may best be displayed graphically rather than textually.

Children may want to get replies in terms of cartoons which they can understand rather then text or graphics which they find difficult to interpret. Users may want to get replies in different forms depending on the time of day, e.g., they may want to get short, verbal summaries late at night but more lengthy, printed information in the early evening. It is desirable that a query language provide the facilities to customize the replies given to a user dependent on factors such as those cited above.

DYNAMIC CONTROL

A user should be able to control the pace at which the reply to a request is presented to him. That is, given that the reply will be presented in a certain form via a certain medium, the system should provide the user with the means for dynamically controlling the presentation. The user should be able to control the speed at which text is presented on a display. He should be able to control this speed dynamically skipping over parts or going back in the text. He should be able to change the form of the reply, say from text to voice, dynamically without reformulating the request. In addition, these changes should be done fairly easily.

REUSABILITY

Often the reply to a request is not only for immediate consumption, but may be useful at a later time or as input in the formulation of another request. It is therefore desirable that the query language allow the reply to a request to be saved for further processing or to be redirected as input to another request. In this way, it is possible to break complex queries into simpler "intellectually manageable" units and to reduce the total amount of interaction between the user and the system. This manner of interaction may more closely approximate the way in which people handle complex tasks.

2.3 Dynamics

We identify five characteristics of the interaction between the user and the system which are important for evaluating interactive query languages.

BANDWIDTH

The bandwidth of interaction is a measure of the speed at which the user and the system communicate with each other. For example, it is usually the case that people can absorb more information if it is presented in pictorial form than if it is presented as written text. Typing a request to the system has a much lower bandwidth than being able to communicate the request verbally. Bandwidth is a function of the technology available for interaction as well as the way in which requests and replies are specified. It is desirable to *maximize* the bandwidth of communication between the user and the system via the query language.

GAMESMANSHIP

It is important that the interaction between the user and the system be interesting to the user. By this we mean that the nature of the interaction is challenging to the user and that the degree of difficulty of interaction is related to the benefit of the outcome. We call this parameter gamesmanship. Users of videotex systems have no external motivation, e.g., monetary, for interacting with the system. The

motivation is purely internal, e.g., they want to learn or be entertained. The user must perceive that the investment of his time is worth the result obtained or at least that the time spent was enjoyable and not frustrating. Thus if formulating requests is interesting and challenging, then the tediousness of the interaction may be alleviated by its learning or entertainment value.

PROTOCOL

A protocol is a specified way in which a user interacts with the system. For example, a user may always formulate a request in the same way by typing it on a keyboard. It is desirable that a query language allow multiple protocols for user/system interaction. In this way, the user can choose the protocol desired at any particular time. It may be desirable to have different protocols for different levels of difficulty of requests. In this way simple requests are easy to formulate, e.g., push a button, while more difficult requests require more effort.

RESPONSIVENESS

Ideally, the system should always respond immediately to a user's request. However, this is not always technically possible. In the absence of this ideal, the responsiveness of the system should be related to the nature of the interaction. Thus, if a request is short or very easy to formulate, then the user should expect that the system can respond to the request quickly. Conversely, if the request is long or very difficult to formulate, then it is not unreasonable to expect to wait longer for a reply. It is desirable to match difficulty in formulating a request with difficulty in processing the request. In this way, the user is not frustrated when he has to wait a long time for a reply to a request which he perceives as being very easy to formulate.

CONTROL

People usually like to feel that they are in control of a situation or at least to know that something they trust in is in control. They do not react favourably when they feel that they are being driven, especially if the driving force is a machine. In interacting with a computerized system, the user should always feel that he is the one that is in control and that the system is his friend. Even though the system is always in control ultimately, it should make the user feel he is in control. The responsiveness and helpfulness of the system are some determining factors that contribute to a feeling of control. The amount of interaction is also important. One feels much more in control in a car where one is constantly doing something to guide the car, than in a bus where one is a passive participant. Thus, receiving feedback from the system as one formulates a request can contribute to the feeling of control. This sense of control may be the most important factor in user acceptance of a system.

2.4 Summary

We have outlined several desirable characteristics of a query language. Sometimes the requirements for desirable characteristics mesh nicely. For example, bandwidth can be maximized by minimizing keystrokes per unit time. At other times the requirements are conflicting. For example, providing many protocols in a query language may sacrifice uniformity since different commands may be used in different protocols. The goal then in a query language design should be to determine and to evaluate the tradeoffs among the different requirements to come up with an appropriate choice of the user

interaction parameters. Rather than choosing one value, it might be desirable to provide several versions of a query language in which different requirements are emphasized. Human factors evaluations can be helpful in deciding on appropriate values for parameters and appropriate approaches.

As an example of the evaluation of user interaction in terms of these requirements, consider requests for stock quotations or weather reports. One of the ways that we can provide the service is to have a complete stock or weather information report available on request. The person pushes a software button "stocks" or "weather" and the system responds with the standard stock or weather report in text form. Let us look at the user interaction parameters to evaluate the desirability of such a service.

In terms of formulating the request, keystrokes are minimal, the command is simple, bad requests cannot be formulated and this type of command can match many applications (by using different buttons) and can be used for many applications. However, the command has no selectivity at all. The user will always get a full stock or weather report. He can not isolate the business sector, company or time in terms of a stock report, or the geographic area or time in terms of a weather report. He gets everything and the onus is on him to pick out the relevant information from what is displayed.

In terms of getting a reply the presentation of the answer is fairly easy to read, the user can scroll the answer with any speed he wants and he can presumably save the reply for future reference. However, the user normally does not have a choice about via what medium or in what format he will get the information. Neither can he match the reply to his terminal's special capabilities, e.g., grahics if they exist. He gets the answer is textual form because it is the common denominator for all output devices. However, a child and a successful businessman may want different output for weather, i.e., the child in terms of a cartoon, while the businessman in terms of a terse factual report.

Consider now the dynamics of interaction. They are virtually non-existent. The request is formulated by pushing a button. This has very high bandwidth in terms of the speed with which the user's request is communicated to the system, but thereafter the interaction is over as far as the user is concerned. He has no control over the answer; he is completely passive. It is a very sterile and boring environment in which to select information.

3 QUERY LANGUAGE EVALUATION

In this section we categorize query languages into seven types. For each type we briefly describe the form of the languages. We then discuss the language type in terms of the user interaction parameters outlined in Section 2.

3.1 Keyword

Keyword query languages allow the user to use a restricted form of his native language to interact with the system. There are numerous examples of these types of languages among them SQL [Denny, 1977], ILL [Lacroix and Pirotte, 1977a,b] and NUL [Deheneffe and Hennebert, 1976]. Requests are formulated in a restricted, English-like grammar where specific words (keywords) signal the start of specific parts of a request. To make the processing of these languages easier, they are very rigid in their syntactic form. Constructs in the language must appear in a fixed order and in a fixed format. Replies to requests are usually presented to the user in a tabular format.

To formulate a request, quite a bit of keystrokes are required on the part of the user. However, once a request has been formulated it is quite easy to read and to interpret since the structure of these languages closely follows English sentence structure. There are usually only a few commands available to the user, but they are quite complex, allowing many options of specification. Thus, it normally takes some time to master an entire language completely although subsets of it may be learned quite quickly [Reisner *et al.*, 1975; Reisner, 1977]. Because of the complexity of each command and the amount of keystrokes required, it is not difficult to form syntactically incorrect requests. However, these are caught immediately by the system. It is possible to form semantically meaningless requests, but difficult if the request is syntactically correct and if the data base has been designed properly. The selectivity of keyword query languages is very high. A user can specify very precisely the data he wishes to select. The languages make no assumption about the nature of the data base in terms of application although they usually assume a certain overall structure to the data, e.g., relational. Thus the languages are application independent, but data base type specific. Keywords in a language can be customized for a given application. Other aspects of a language are rigidly fixed.

Replies to requests are usually presented to the user in a tabular format. The user has no control over the form of the output. Although tables are fairly easy to follow for some types of output, they can be very difficult to interpret if there are complex relationships between tables. Output is normally presented only on a screen or a printer. Customizing the output for the user is not provided for by the languages. Output is usually presented with table and column headings and the user may be able to customize these. No dynamic control over the output is provided other than that provided by the nature of the display device itself, e.g., scrolling. Most keyword languages will allow the user to save the result of a query and/or use it for further processing.

The dynamics of interaction for keyword query languages are very poor. The bandwidth of interaction is very low. Requests must be typed in and replies must be read from a screen or printer. There is virtually no gamesmanship involved in the interaction; everything is very cut and dry. There is only one protocol for interacting with the system: typing in the format required by the language. In some keyword query languages requests that are difficult to process, e.g., joins in relational systems, are quite simple to specify in the language. Thus the responsiveness of the system does not match the complexity of request formulation. Because of the limited way requests are formulated, the system is not able to provide much feedback to the user until after the entire request has been specified. In such

an environment there is not much feeling of control over the system by the user.

3.2 By Example

By example query languages are ones in which the user constructs his requests by giving the system an example of a reply to the request. The example is constructed according to the format for replies used by the system. For example, in Query-by-example [Zloof, 1975a,b; Zloof, 1977, 1980] templates of tables are displayed on a screen and the user fills in the table according to which columns he wants in the reply and the kinds of values desired in each column. In TLA, templates of forms are filled in to specify a query [Hogg, 1981; Hogg *et al.*, 1981; Nierstrasz, 1981]. Similar languages have been proposed by other researchers [Hammer *et al.*, 1977; Luo and Yao, 1981].

In these types of query languages, while some input is required from the user, the keystrokes required are quite small. The system supplies a great deal of the input, such as attribute names, that would be required in a keyword query language. The user need only indicate which attribute values he wants in the reply (usually involving a single keystroke) and any conditions on the selection of values. Commands indicating the operation desired are usually limited to one character codes. The commands in these languages are few and simple. However, they are not well suited for specifying complex queries that span several templates. The readability of the request is compromised somewhat for simplicity of expression. It is very difficult to formulate syntactically incorrect requests because of the rigid form of the specification. For this reason semantically incorrect requests are also difficult to form. The user always is aware of the information the system has since it is displayed to him in the templates and only requests about this information can be formulated. The selectivity of these languages is comparable to that of keyword query languages. The way in which the queries are specified is independent of the application; only the templates change, not the manner in which they are filled in. As long as the data base that is being accessed can be represented by the form of the templates used in the language, then the language is also data base independent. Customizing of the language for an application can only be done in terms of attribute names within the application. Because of the rigid language format, it is difficult to take advantage of any special characteristics of an application.

Replies to requests are presented to the user in terms of the templates used to formulate the request. Thus there is a one to one correspondence between requests and replies. In addition, since these are usually in the form of tables or forms, they are fairly easy to follow and understand. However, this characteristic also limits the reply in that it must be presented on a screen or printer. In the same way that customizing of the requests is difficult, customizing of the replies is also difficult. Their form is determined by the form of the request. Because of the nature of the replies, dynamic control is not possible except for screen scrolling. These languages usually allow the user to save the result of a request and to use it for further processing.

In terms of dynamics, we have a request-reply environment in which the two processes are disjoint. The bandwidth is fairly high in terms of time of specifying requests and is related to the complexity of the request. Simple queries can be specified quickly, while more complex ones require some effort. The output bandwidth is limited by the speed at which a user can read a screen. These types of languages have a somewhat higher degree of gamesmanship than keyword query languages. Specification is less tedious and in some systems the user is guided by the system in filling a form or table, i.e., the cursor can skip automatically to the next attribute rather than the user having to space to it. On the other hand, the user is constrained to always specify requests by filling in templates. In terms of responsiveness, we have already mentioned that some queries, e.g., those that go across

templates, are difficult to formulate. This difficulty of formulation also corresponds to those queries that are difficult to process. Because of the request-reply nature of the environment, the degree of control that the user has over the system is not high. Some guidance can be provided by the system, but the helpfulness of the system is limited.

3.3 Natural Language

Natural language query languages attempt to allow the user to specify requests in his native language. The goal is not to limit the vocabulary in any way, although in practice only a subset of a language can be processed by the system. There are many examples of such systems developed by researchers in artificial intelligence [Woods, 1973; Mylopoulos *et al.*, 1975, 1976; Harris, 1977; Hendrix *et al.*, 1978]. Perhaps the best known example that makes use of data base technology is the RENDEZVOUS system developed at IBM San Jose Research Labs [Codd *et al.*, 1978].

Requests to such systems are formulated in terms of a dialogue between the user and the system. This means a very high number of keystrokes are required to specify a request. There are no fixed commands *per se*. Like for English sentences, simple requests are fairly easy to specify while complex ones are difficult. Because the user has complete freedom to use the entire English language, the same request can be specified in many ways. This means that there are many ways in which a request can be formulated badly and misunderstood by the system [Shneiderman, 1978]. Syntactically, these systems usually provide spelling correction features. These languages can be very selective, but this depends on how precisely the user specifies his request. Such specification can be very difficult since the language provides no guidance in this matter. Natural language is the most uniform language one can have for specifying requests. It is uniform across applications as well as across data bases. In addition, the language can be customized by the user according to the application simply by using his knowledge about the application. This assumes that the system also has some knowledge about the application so that it can understand the vocabulary of the user.

Replies to requests can be presented in terms of English sentences for single facts. For many facts, this could pose some difficulty. Usually replies are presented in terms of typed sentences on screens. However, there is the possibility of adapting the sentences to voice output. The form of the output, as English sentences, allows for its customization. Words specific to an application or to the type of audience can be used. Because of the dialogue nature of the interaction, the user is able to exercise dynamic control over the reply to a request. The user can interact with the system until the reply contains the information of interest to the user. Reusing and saving the replies is difficult unless the system has the capability of digesting new facts that are presented as natural language sentences.

The dynamics of interaction consist of a dialogue between the user and the system. For typed input, this has a very low bandwidth since entire sentences must be typed. If the output is also in terms of typed sentences, then this bandwidth is also very low. Since the user is carrying on a dialogue with the system, the system can incorporate some gamesmanship into the interaction by the way in which it responds to the users query. For example, it can ask the user to clarify parts of his request that it does not understand. It can also rephrase the user's request in different terms. This also allows multiple ways for the user to interact with the system, e.g., by stating sentences, by answering the systems questions or by making choices supplied by the system. Because requests are formulated in a piecewise fashion, the user always has the illusion that he is making progress in satisfying his request. This also allows the system to process complex requests in a piecewise way so that the delay between request formulation and response is minimized. The piecewise approach to request formulation also

makes the user feel that he is in control since he is guiding the system to the answer.

3.4 Menu

Query languages that are menu-based allow the user to specify requests by pointing at the action or data that is desired. At any point in time, the user can choose from a menu of available operations or data objects. Choosing an operation or object may cause additional menus to be made available. In this way, requests are specified by steps in a hierarchical manner. These types of systems have been used extensively in many applications. The Officetalk system at Xerox Palo Alto Research Center is an example of a system that uses menus for querying data bases [Ellis and Nutt, 1980].

Requests are specified by pointing at desired actions. Thus, a minimum number of keystrokes are used. The commands available are simple and represented by either English keywords or by icons that represent the action or object of interest. For example, to store a document on the screen one can point at the document and then at the image of a filing cabinet. Because of the hierarchical structure of menus, the commands are adequate for simple requests that are hierarchically structured, but inadequate for more complex requests. The highly structured nature of the interaction makes it difficult to formulate incorrect requests. The interaction is completely under the control of the system at every stage. Selectivity is very good for queries that have been anticipated, i.e., those that follow the menu structure, but more difficult for other requests. The same technique of menu selection can be used across applications and even across data base types. Menus can be customized to specific applications by changing the keywords or icons that are used.

Replies to requests are presented to the user in the form of the objects that are stored by the system. In most systems employing menu techniques these are usually forms containing structured text or a table. Thus replies presented in this way are fairly straightforward to follow. Output is normally available only on a screen or a printer and must be read by the user. If the system allows definition of new objects on the fly as in Officetalk, then the output can be customized for the user. However, this customization is only in terms of the types of objects that are maintained by the system, e.g., forms. The user can dynamically control the presentation of the output if he is allowed to use the menu operations as the output is being presented to him. Thus, if a reply has several forms as the answer, then he may be able to look at each one in turn and take some further action upon it before looking at the next one. Facilities may be available for saving the result of a request and/or using it for further processing.

The dynamics of the interaction in these types of languages are somewhat better than in previous languages. Input bandwidth is reasonably good once the mechanics of the pointing operation have been mastered. This bandwidth increases if screens that allow multiple windows are used. Output bandwidth is somewhat restricted because of the necessity for reading the reply. The pointing type of environment has interesting possibilities for gamesmanship. It is certainly much less tedious than keyboard interaction and the use of icons to represent actions can help maintain user interest. In the Officetalk system the user can interact with the system in different ways. For example, he can switch to an editor to edit input and output, or to a drawing mode for free-hand sketches. Because the pointing operation on the part of the user has high bandwidth, the user may expect that all operations are equally complex. However, this is not usually the case. Thus processing speeds and response time may not be uniform which may frustrate the user. The nature of the interaction and its high bandwidth give the user the illusion that he is in control, always directing the system. However, this may not be the case for queries that do not follow the structure of the menus.

3.5 Graphic

Graphic query languages formulate requests mainly in terms of diagrams, but some typed input is also usually required. Each type of object in the data base is assigned a specific geometric shape. The user then manipulates these shapes to specify requests. Special conventions are used to specify selection of specific data or relationships between data. Two examples of these types of languages are the LSL graphic language [Lipson and Lapczak, 1976] and the CUPID language of the INGRES system [McDonald and Stonebraker, 1975].

Although keystrokes are minimal in formulating a request, quite a bit of work is required by the user to construct the diagrams that represent a query. Geometric objects have to be manipulated on the screen and processing options specified for some of them. Once the form of the request has been specified by a diagram, the action desired is fairly easy to invoke. The commands therefore are simple; the complexity is in constructing the diagrams. Syntactically incorrect requests are difficult to formulate since the way in which objects can be put together and connected is controlled by the system. Similarly, semantically incorrect requests are difficult to formulate, although it is possible. Graphic query languages can have the same degree of selectivity as keyword languages since they employ the same types of constructs only specified differently. The languages can be used across applications that use the same type of data base, since the data bases employ the same types of objects for representing data. Across different data base types, these languages may not apply unless conventions concerning the interpretation of geometric objects in different contexts are specified. Customizing for applications can be done in terms of the data object names used and keywords employed in the language.

Replies to requests are normally presented in terms of a tabular structure much like for keyword query languages. Therefore, replies have much the same characteristics in terms of presentation, multi-media, customizing, dynamic control and reusability.

The dynamics of interaction for graphic query languages are not particularly good. As a communications medium, pictures have a high bandwidth. However, the bandwidth achieved in formulating requests is not very high. It takes much time and effort to formulate requests via diagrams. It may be easier to specify the request by some other mechanism and use diagrams to display the specification, i.e., the system constructs the diagram, not the user. Diagrams would appear to have a high degree of gamesmanship. It is fun to draw pictures, but only if it can be done easily. Currently this is not the case with these languages, particularly for complex requests. In addition, it may not always be clear how to formulate a request in terms of a diagram, especially for naive users. Formulating requests graphically may be a long process even for simple queries. Thus the user gets no real sense of how complex a query is from its formulation. Also in terms of perception, the process of specifying requests is slow and will seem slow to the user because of limited interaction with the system. It is doubtful that the users of these languages feel that they are in control of the interaction.

3.6 Multi-media

Multi-media query languages, as their name implies, use many different ways of interacting with the system. Text, formatted data, voice, graphics, colour, animation, etc. are used to formulate requests and to present replies. Categories of data in the data base are represented by icons. This categorization can be specified at several levels of abstraction. Users can browse through the icons at a particular level and select one for further processing. The effect of icon selection is to "zoom in" on the data in question and to be presented either with further icons containing more detail about the category of data,

or with the actual data itself. Icons can represent textual data, formatted data, visual data, etc. The most advanced work of this type is the spatial data management system developed at MIT and CCA in Boston [Herot, 1980].

Requests can potentially be formulated in many ways, e.g., by pointing at icons, by typing statements, by selecting from menus, etc. The same data can be selected in different ways depending on the mood of the user or his knowledge level about the data base. The commands used can be as simple as pointing or as complex as keyword query language specification. Because of the flexibility involved in formulating requests, there is the possibility of looking in the wrong place for data. This is especially true if one is in a browsing mode. The degree of selectivity possible depends to a large extent on the construction of the data base. That is, if there are many levels of abstraction between the highest level and the data, then the "zooming in" process can be fairly selective. Otherwise, a great deal of irrelevant data may be selected. Of course, the user has the possibility of using a more selective query language at any time. These techniques can be used across applications and across data base types. However, the particular technique used, e.g., pointing, may be more appropriate in some situations than in others. Customizing for applications is done in terms of the type of data that is stored about an application, and how it is stored, at each level in the abstraction of a data base.

Replies to requests depend on the level of abstraction one is at and the nature of the data base. It can be in terms of text, pictures, graphics, etc., whichever is the most appropriate representation for the data. This allows data to be stored in a way that is appropriate for the intended audience. The user can direct the system to display the reply in a certain way or via a certain medium. The user has complete control over how the output is displayed, where, in what form, and at what rate. Although the results of a query can be saved, it is not always possible to use it as input to another query, e.g., if it is in graphical form or verbal form.

In terms of dynamics of interaction, such a system is excellent. Input and output bandwidth is high and the user can choose the form depending on his tastes. The interaction has a high degree of gamesmanship. The user can "fly" over the data and zoom in on a particular part very easily. Interaction with the system can be done in several ways and can change as a request is formulated. Response time of the user can be very fast to a request by the system. However, the system's response time to a user's request may be quite slow because of the interaction of different technologies involved in the interface, at least at the present time. Thus the interaction may become frustrating to the user especially if response time is not uniform for similar requests. The perception of control by the user can be quite high in such a system. The user has available many ways of directing and instructing the system. The system can be helpful in guiding the user interaction by monitoring the nature of the interaction. For example, it can provide more help if the user is making many mistakes, or it can provide fewer levels of abstraction if the user is following a well worn, to him, path.

3.7 Computer Games

There have been many computer games developed for accessing useful information or for amusing researchers. Most of these games are not generally known because they operate on very specialized data bases in localized areas. For example, there is a game at Bell Labs called EATS for finding a restaurant of a certain type in a certain location. The idea behind these games is that you search for something by playing a game with the system. For every input you supply, the system tells you something about the nature of your current environment. From this one can tell whether one is near or far from one's objective. Techniques used for interacting with such a game are for the system to ask

questions of the user or for it to supply the user with some choices for actions.

Requests in such an environment are specified incrementally. For example, in a question-answer type of game, the system asks a question, the user supplies an answer and the system tells the user something about the current status of the game. This type of request formulation can involve several keystrokes in supplying answers. However, answers are usually short and may merely involve a yes-no response. The commands are very simple and once initiated, the system guides the user so that it is impossible to formulate syntactically incorrect requests. However, semantically one can get into difficulty because responses to questions are made in the context of limited information about the current environment. Thus one could easily head off in the wrong direction. If one keeps a straight path to the desired information, then the selectivity is good. However, if one strays, then a great deal of useless information may be obtained. Unfortunately, the nature of the request formulation is very application dependent. On the other hand, the techniques used can be very insensitive to the type of underlying data base. The interface can be easily customized to particular applications.

Replies are presented to the user in a piecewise manner. Usually parts of the reply are presented to or discovered by the user as the request formulation proceeds. Thus, the presentation is usually very easy to follow since one is presented with only a few facts at a time. Replies are usually in terms of screen output, but sound can be used to indicate that one is nearing an objective or is on the right track, e.g., as in T.V. computer games. The form of the response can be customized to the intended audience as in T.V. computer games. The user can control the rate at which he obtains the output by controlling the rate at which he responds to the system. Because of the nature of the interaction, saving previous replies and reusing them can be difficult.

In terms of dynamics, the interaction can be a lot of fun for the user. Input and output bandwidth can be fairly high especially if input requirements are minimal, e.g., yes-no responses. The interaction obviously has a great deal of gamesmanship in it. The user is playing a game in which he is trying to win something. For each game there is normally only a single protocol of interaction, but this protocol can be different for different games. Because of the step-wise, short interaction, response of both the system and the user to each other can be quite fast. System response should be uniform since at each step only a relatively small number of possiblities arise (especially if yes/no answers are expected). The perception of who is in control of the interaction depends on the users perception of whether or not he is winning. If the game is designed properly so that it is not too difficult to win, then the user will feel that he is in control.

4 QUERYING REQUIREMENTS

In a previous section, we divided the querying process into three parts: request, reply and dynamics of interaction. As far as the user interaction is concerned, the most difficult of these three parts is the request part and its dynamics. The reason for this is that in forming a request the user is trying to inform the system precisely of what he wants. The user must provide the system with a great deal of guidance and be very exact in specifying a request if the system is to understand it correctly. Replies from the system on the other hand are much easier for the user to understand even if they are presented badly. This is because humans are much more intuitive and adaptable than computer systems. This is not to say that reply aspects of querying are not important. However, for the rest of this report we will concentrate mainly on the request aspects of a query language and its dynamics.

For both the request and dynamics aspects of a query language, certain characteristics were identified that described the parameters of the user interaction. From these characteristics and their evaluation with respect to different query languages, we can derive certain desirable characteristics for a query language for external data bases.

In general, the user wishing to access the external data bases is not a typist and probably does not want to become one to use the query language. Hence, there should be a minimum number of keystrokes required to communicate with the system. In order to meet this requirement it is essential that some type of pointing mechanism be available for pointing at things on the screen. The pointing method should be mechanical, e.g., a joystick or mouse, so that the user is free to concentrate on the screen while manipulating the pointing device. Most likely, it will be impossible to eliminate all typing. Therefore, to be least tedious, it is also essential that typing mistakes be allowed and corrected by the system and that the user be able to assign aliases to refer to objects in the external data bases.

Any commands that are required to direct the system to perform some action should be provided via function keys or menus of operations and be named to reflect their functionality. In addition, the number of such commands should be kept to a minimum. In this way the user is not required to remember the commands available and can easily recall their function from their names.

The system should guide the user in formulating his requests. To this end it should display options available and also provide instructions on-line at any point during an interaction. The way in which requests are formulated should be intuitive to the user. In this respect, there should be a paradigm for formulating requests that the user can easily understand and that helps him remember the interaction protocols. In addition, the system can provide cues to the user, e.g., by the form of the display, that help the user formulate correct requests.

While providing guidance to the user in formulating his requests, we must be careful not to put the user in a straightjacket by completely predetermining the requests that can be formulated. The query language must still allow the user some latitude in determining the selectivity of his requests. To this end, the query language must at least allow the user to specify the contents of his reply, if not its exact format.

In most query languages the user is supposed to know the structure, called the schema, of the data base he is accessing. For external data bases, we cannot expect the user to know or even understand the structure of the data base or the difference between the schema and the data base. Instead, there should be a mechanism to guide the user through the structure part to the data part. In addition, the way in which the user queries the structure of a data base should be as similar as possible to the way he queries the data base itself. In this way the user only needs to learn one set of interaction protocols that apply to all his interactions with the system.

Studies have shown that users of computer systems can become very lost if they do not know

where they are in the system [Mantei, 1981]. In addition, a user cannot be expected to remember everything that has transpired during an interaction. To help orient the user, we should retain on the screen as much as possible the route we have followed in querying the data base (or at least have it available for display).

User interfaces for querying nonformatted data are often very different from those that query formatted data. When querying external data bases, we may have many different types of data available. Most data will probably be in such a form that a user cannot distinguish formatted data, e.g., inventory information, from nonformatted data, e.g., catalogue sales information. In addition, it is probably not desirable to require a different set of protocols for different types of data. Thus, the user interface should be as uniform as possible for querying the different types of data found in external data bases.

Even though we provide uniform interaction protocols for all types of requests, we can allow a very nonlinear user interface in terms of difficulty of request formulation. That is, simple requests are extremely easy to formulate while more elaborate requests can be very complex. In fact, joins and other complex requests can be assumed not to be frequent. In any case, we will suffer in performance when we do them. The user interface should allow these queries through a higher level, expertise interface which can access the same data as the interface for naive users.

Finally, the user interface should use to advantage the ability of people to remember abstract patterns. In addition, the available technology (graphics and colour) should be used to distinguish between different types of things on the screen. For example, we should distinguish between the data that the user is searching for, is providing for selection or is provided by the system. Although we will not elaborate further on this aspect in this report, it can be very important for guiding the user in his interactions with the system and should be carefully considered in the design of the screen layout presented to the user.

With these considerations as our guiding principals for request formulation and user interaction, we will outline a design for a query language for external data bases in the rest of this report. We first present a paradigm for the query language that conveys by an easy to understand example the essence of the user interaction. We then present an overview of the query language design. Finally, we discuss the design considerations for the query language.

5 PARADIGM

For accessing information via an interactive query language for external data bases the user should have a paradigm that he or she can follow and easily understand. This paradigm should help the user remember (and learn) the querying protocols. The paradigm that we choose is that of navigating a personal spaceship through space.

Our spaceship is of the latest design and is very powerful. It can travel both long distances, e.g., those between galaxies, and short distances, e.g., those between nearby planets, with relative ease. Our usual objective in travelling through space is to arrive at a planet and explore it, although we may just want to explore space per se. Once we arrive at a planet, we can use our spaceship to explore it by flying around or by landing and examining an area in more detail.

Space is very vast and complex, and usually unfamiliar to us. Our spaceship is equipped with a view screen which can show us a visual picture of space in any direction, but only a limited portion of it at a time. It is very difficult, in general, to navigate without aids, i.e., visually only, although when we are exploring an area of a planet this may be desirable. To guide us in our travels, our spaceship can provide us with detailed maps. These maps come in different levels of abstraction from those showing the general structure of a galaxy to those showing the general structure of a planet. These maps can be displayed on the view screen in the spaceship.

We can navigate our spaceship in one of several ways. If we know where we want to go, we can specify the destination and our spaceship can automatically pilot us to the destination. On the other hand, if we are just exploring space, then we may not know our next destination specifically. We may have some idea of the type of destination we want to arrive at, or we may just be interested in looking around space. To learn something about the potential destinations we can look at the maps of them on the view screen. We can ask to look at either specific maps having a certain property or properties, or at all maps. In general, as we explore possible destinations we will look at several maps from very general ones to more specific ones, e.g., from galaxy to star system to planet. Eventually, we will choose a destination and direct our spaceship to take us to it.

In a similar fashion, we can think of the information sources that we wish to access as residing in a vast *information space*. The composition of this information space is patterned after the composition of space itself. It is composed of abstract objects, such as galaxies and star systems, and concrete objects such as planets. In the case of the information space, the abstract objects correspond to the *structure part* (schema) of the data. The structure part tells us something about the nature of the data we can access. The concrete part corresponds to the *data part* (data base) of the information space. The data part is the smallest level of detail that we can explore in the information space.

Using our spaceship (the computer system) we can explore the information space. Our spaceship command post in this case is provided by the query language and our view screen is provided by the CRT (TV) screen through which we view the information space. Just as we can only see a portion of space at a time on our spaceship view screen, so we can only see a portion of the information space at a time on our CRT screen. The CRT screen defines the boundaries of the information space that are currently in view as our spaceship view screen defines the part of space that can currently be viewed. We will call the CRT screen our *portal* on the information space since it acts like a view port allowing us to see only that information currently in view through the view port.

In order to manipulate and maneuver in the information space we require three basic actions, much like for maneuvering through space. First, we need to be able to display in our portal the appropriate structure parts (maps) of interest for the information space so that we can select our destination. Second, we need to be able to go to our destination, that is gain access to the information source of

interest. Finally, we need to be able to explore the data in the selected information source.

We note two points about this paradigm. First, we distinctly separate the structure part from the data part conceptually, but operationally (as we will see) they are handled (almost) the same. This separation could be important in the future as the structure part could reside in the user's terminal much like the maps in the paradigm reside in the spaceship. The data part exists at an information source and to access and explore it the user has to "go to" the information source just as he has to go to a planet to explore it in detail. Second, the way in which we have described the user interaction with the system, most of the details of how to access information are automatic. The user supplies certain guidance, but the system worries about the details of how to do things. This mode of operation is preferred for the novice user, but may be tedious for the experienced user. Therefore, we could provide a manual override to the user and let him guide the system more directly, much like we could provide a manual override in our spaceship. This corresponds to the option of providing a more powerful and complex query language for the experienced user. In this report we only discuss the "automatic" query language, but the possibility of a "manual" query language as a complement to the former should not be discounted.

6 DESIGN OVERVIEW

In this section we will describe the overall design of an interactive query language for accessing external data bases. We will discuss *what* can be done and not *how* to do it. Thus we will describe the operations generically and use generic terms such as "point at", "fill in" and "select" without discussing how these operations are done. We will use the analogy developed in the previous section to explain the operations in the query language. In the next section, we will consider some of the ways in which these operations might be done by the user.

6.1 Choosing an External Data Base

Suppose that we are in our personal spaceship somewhere out in space and would like to choose a destination to go to. Using our view screen, we can scan space and look at the stars. However, all that we see by doing this is an abstract pattern. If we are experienced space pilots, we may be able to choose a destination from the star pattern directly and direct our spaceship to go to it. However, in general we probably will not be familiar with space. To choose a destination for our spaceship, we can scan different parts of space and look at maps of each part. We can ask to look at only certain parts of space that have some specified properties or we can look at all parts of space. As we look at the maps, we can ask for a more detailed map of a specific region of the current map. While looking at the maps, we may mark some regions of a map for future reference as being of interest and requiring a second look. Eventually, we will decide on a destination and direct our spaceship to go there.

Consider now trying to access external data bases. When we first sign on, we are like the person in the spaceship. We are out in space and in the portal we can see an abstract pattern (template) of the data bases that we can select from. The image that we see is like the view of space that just shows us the star patterns without any information about what they represent. For example, we can be presented with a template in the portal such as that shown in figure 1.

The exact format of the template in the portal is not important here (this is subject to study to determine the best layout). The important point to note is that the template is divided into three basic

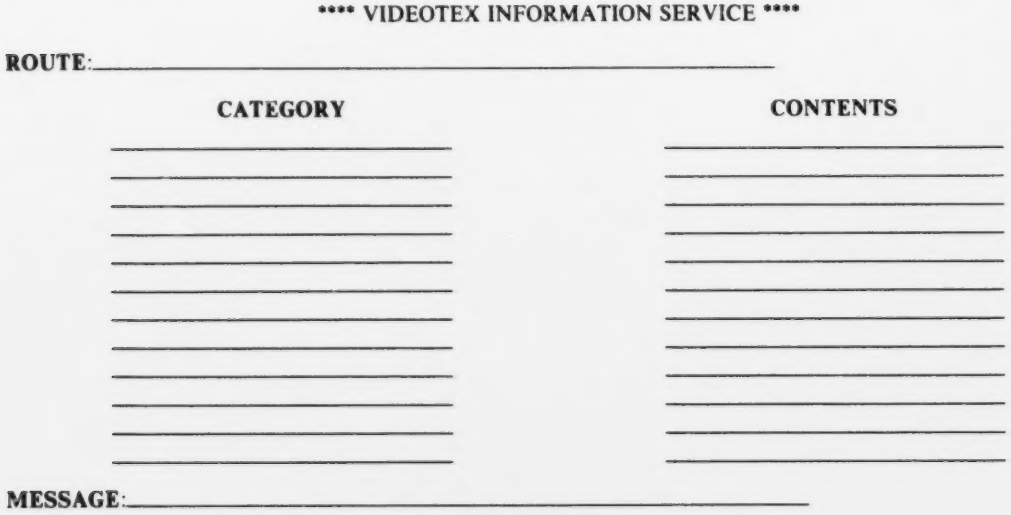

Fig. 1 Example template.

**** VIDEOTEX INFORMATION SERVICE ****

ROUTE:_____

CATEGORY	CONTENTS
Entertainment	Films
	Live arts
	Exhibits
	Sports
	Television
News	World
	National
	Local
	Sports
	Entertainment
	Weather
Restaurants	Chinese

MESSAGE: Frame 1 of 10

Fig. 2 Example display.

parts. One part specifies the *path* we have taken to arrive at the current template in the portal. The **ROUTE** field in the example contains this information. A second part is used for the *specification and display* of data. In the example the **CATEGORY** and **CONTENTS** fields serve this purpose. Finally, there is a part used for *system feedback* to the user, labelled **MESSAGE** in the example.

If we have no idea which external data base we would like to access, we can ask the system to display all the information it has about data bases at this level. In this case we get a display of all entries that correspond to the template. For example, we might get a display such as that shown in figure 2 if we initially ask for a display of all entries.

We now can examine the information map, marking some entries for future reference by pointing at them or selecting one by pointing at it. Selecting an entry allows us to look at more detailed information about it. For example, selecting the category 'entertainment' and requesting a display of all its entries would give us a display such as that shown in figure 3.

Alternatively, we may know the kinds of external data bases we are interested in, but not where

**** VIDEOTEX INFORMATION SERVICE ****

ROUTE: Entertainment

CATEGORY	CONTENTS
Films	First run
	Revues
	Festivals
Live arts	Theatre
	Ballet
	Opera
	Recitals
Exhibits	Art
	Books
	Crafts
Sports	Hockey
	Soccer

MESSAGE: Frame 1 of 2

Fig. 3 Example display.

**** VIDEOTEX INFORMATION SERVICE ****

ROUTE: _____

CATEGORY	**CONTENTS**
Entertainment	Films Live arts Exhibits Sports Television
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

MESSAGE: Frame 1 of 1

Fig. 4 Example display.

they are. In this case, we can ask the system to show us more information about these destinations by pointing at and filling in the **CATEGORY** field. For example, we may fill in the value 'entertainment' in the **CATEGORY** field and ask for a display of its contents as in figure 4.

Subsequently we can examine the information map, mark some entries for future reference and/or select one by pointing at it. For example, if we select the 'entertainment' entry and subsequently ask for a display of its entries we get the display shown in figure 3.

Rather than explicitly entering a field value, we can fill in a field value by pointing at the field and requesting a display of all the field values. We can then mark some entries as being the values we want to fill in for this field. In this case rather than entering a value explicitly, we do it implicitly by marking some values from those displayed. Thus, for example, we could specify the display shown in figure 4 by pointing at and requesting a display of the values in the **CATEGORY** field, marking the entry 'entertainment' and requesting a display again.

Finally, if we know where we want to go, then we can specify this directly and the system will take us there. We specify this information by pointing at and filling in a value for the **ROUTE** field. This directs the system to a certain destination and causes it to show us a template of the destination selected.

By applying this technique repeatedly, we can navigate the structure of a data base to arrive at the data of interest. Every time we select something in a template we get more detailed information about the route to a data base. We also retain information about the path we chose to get to the current template. This is analogous to trying to determine a destination for our spaceship. This technique can be carried out to several levels, but it is desirable to keep the number of levels small to reduce the amount of interaction required to get to the data. In general, the number of levels required will be determined by the complexity of the data base being accessed. This aspect is in line with our stated goal of relating complexity of access to complexity of the information or type of request formulated.

By means of the **CATEGORY** and **CONTENT** fields, we in effect provide a keyword and cross referencing capability. That is, all data bases related to a specified category or categories will be displayed. As well, some information may apply to more than one category. Note that the displayed information may be at different levels in the structure part. That is, some may refer to data bases

themselves, while others may refer to parts of a data base. However, this is irrelevant to a user and he need not be aware of this distinction. What appears in the portal is a function of how the cross references to data have been designed within the system.

Associated with each keyword displayed in the portal is a more detailed description of the information represented by the keyword. At any point in time we can point at a keyword and ask for a description of it. This is analagous to consulting a guidebook associated with a map. We only do so when we want to know more detailed information about a point of interest.

The **ROUTE** field allows us to see the path we have taken to arrive at our current location. At any time we can backtrack by pointing at the level (name in the path) we want to go back to.

6.2 Exploring the Data

Our eventual destination in our spaceship is a planet. Once we arrive at a planet, we can fly around and look at it or we can land and explore a small area of the planet. Again to help us navigate, we require maps of the planet. These maps show us the abstract structure of the planet but no detail about it. By flying around and/or landing we can explore the planet in detail.

In a similar manner, we eventually reach the data part of a data base. As in exploring and navigating around the structure part, we can now explore and navigate around the data part. As for the structure part, we again are presented with an abstract pattern (template) of the data in the portal. An example of such a template is shown in figure 5.

**** VIDEOTEX INFORMATION SERVICE ****

ROUTE: Entertainment.Films.First run

FILM	STARRING	THEATRE	TIMES	PHONE#

MESSAGE:_____

Fig. 5 Example template.

Specific template entries are selected by pointing at and filling in a field. For example, to display all films starring Donald Sutherland, we would point at the **STARRING** field and enter the value 'Donald Sutherland'. The system would then display all films starring Donald Sutherland that are showing currently. Alternatively, we could fill in a field by pointing at it and asking the system to display all values that it has for that field. We can then mark certain field values by pointing at them. The system will then show us a complete display of all entries in which that field value appears. For example, we could ask that the system show us all values for the **STARRING** field. We then choose those values of

interest to us for further inspection.

If we are just interested in browsing, then we can ask that all template entries be displayed. Subsequently, we can mark specific entries for further examination by pointing at them. This action has the effect of eliminating those entries that are not of interest to us and thus reducing the size of the data display.

6.3 Predefined Routes

As we travel around in our spaceship, we soon discover that there are certain destinations that we go to very often. We become familiar with the routing to these destinations and soon do not require maps to help us get there. We can specify the destination precisely by visual navigation.

In a similar manner, when accessing external data bases there will be certain data that we access very frequently. To speed up our routing to this data, we can predefine the route that the system should take. We can name this route and then specify it explicitly in the **ROUTE** field. This is analagous to pointing to our destination directly on the visual star patterns that we see from our spaceship.

These predefined routes are appended to the structure part of the videotex information service, but are only available to the user who defined the routes. The destination of the routing can be a point in the structure part, data part or part of the data part, i.e., a subset of the data. In this way the output that the user receives can be customized to some extent by, for example, filtering out certain information in the data part. Predefined route names can be distinguished from regular data base names by highlighting them in some suitable manner.

7 DESIGN CONSIDERATIONS

In the previous section we discussed what the query language can do, but no details were provided about how it might do it. We used generic terms such as "point at", "fill in" and "select" without specifying how these actions might be done. In this section we will elaborate on how some of the actions discussed in the previous section might be done by a videotex user.

7.1 Pointing

One of most fundamental actions required in the interaction protocols discussed in the previous section is the ability to point at something within a portal for the purpose of marking or selecting it for further actions. Ideally, this pointing should not distract the user's attention from what is being displayed in the portal. Voice directed pointing would be highly desirable, but perhaps not technically or economically feasible currently. The next best pointing mechanism would be a mechanical pointing device such as a joystick, puck or mouse. The user is then able to position a cursor to the appropriate position within the portal and to indicate the selection of this position in some way. (Note that it is neccesary to both position the cursor and to indicate that this is the position desired.) Such a pointing capability is technically feasible now and possibly could be provided economically.

In the absence of a pointing device for specifying an arbitrary location in the portal, an alternative is to be able to specify specific locations, namely specific fields in the portal. This specification should be made as easily as possible and should take account of the fact that positioning is a function of the template structure currently displayed in the portal. We generally need to be able to specify a field in a template and perhaps an entry within a field. For generality, these two specifications should be independent of each other. The exact semantics of the specification will depend on the characteristics of the template displayed in the portal.

The ability to select a specific field and a specific entry leads to the notion of a *current* field and entry. Visually this situation can be indicated by highlighting the current entry and field in some way, e.g., blinking, inverse video or a certain colour. Operations can be specified that then apply to the current entry or field as discussed later in this section.

For field specification, there is a **field** operation which has the following options and semantics: **field** *next* — selects the next field **field** *previous* — selects the previous field **field** *name* — selects the named field.

Fields in a template have a predefined ordering which is used by the *next* and *previous* options. Fields also wrap around. The initial field is the entire template as is the field after the last field and before the first field. The *name* option uses a symbolic name to refer to a field directly. Conventions need to be established for naming fields symbolically. Some functions (as defined below) require a field as a parameter. If no field is explicitly specified, then the default field is the entire template. Thus, **display** mode with no field explicitly specified refers to a display of all the values associated with a template, while **display** mode following the selection of a field refers to a display of only the values associated with the selected field.

For entry specification, there is an **entry** operation that allows an entry within a field to be selected with the following options and semantics: **entry** *next* — selects the next entry **entry** *previous* — selects the previous entry **entry** *name* — selects the named entry.
Entries have a predefined ordering which the *next* and *previous* options utilize. Entries can also be referred to by symbolic names which allows an entry to be selected directly. For example, for a list of

values within a field, entries can be referred to symbolically as 1, 2, etc. If the current field is the template, then an entry refers to all the values that define one instance of the template.

7.2 Portal Positioning

In our discussion so far, we have not considered any restrictions on the size of the template that could be displayed in the portal. We implicitly assumed that everything could be displayed at once. In general, however, templates may be larger than the portal size. Thus, the size of our portal limits the amount of information that we can display at any one time. Ideally, we would like to be able to move the portal continuously across the template as if it were a TV camera.

In videotex systems, the information space is usually divided into fixed size units (pages) which are the size of the portal (TV screen) and continuous motion in all directions is not possible. Accordingly, we can assume instead that templates are divided up into fixed size *frames*. A frame is a part of the information space that can be viewed through the portal at one time (much as one views a photographic slide through a slide projector). In general, frames exist in all directions from the current frame (rather than just forward and backwards as when viewed through a slide projector).

A frame contains just that amount of data that will fit entirely within the portal. Frames above and below the current frame hold, respectively, previous and next template entries. Frames to the right and left of the current frame hold additional data related to the current frame that cannot be displayed all at once through the portal. Thus, for example, tables of data that are too wide to fit within the portal can be divided up into several vertically adjacent frames (right and/or left of each other). While tables of data that are too long to fit within the portal can be divided up into several horizontally adjacent frames (above and/or below each other).

To position the portal to the desired frame, there is a **frame** operation with the following options and semantics: **frame** *up* — moves the portal to the frame above the current frame **frame** *down* — moves the portal to the frame below the current frame **frame** *left* — moves the portal to the left adjacent frame **frame** *right* — moves the portal to the right adjacent frame **frame** *name* — moves the portal to the named frame.

Frames can be named symbolically so that it is possible to position the portal on them directly. In most cases, and probably initially, it is desirable not to have right and left frames as this will fragment the data a great deal. In this case the **frame** function reduces to the *up*, *down* and *name* options.

If it is necessary to have left and right frames, then some way of orienting the user as to his current position is desirable. This orientation has two aspects to it. First, we want to tell the user where he is with respect to the entire information space visible at this level. One technique used to do this is to graphically show the position of the portal with respect to the entire information space. For example, we could show the entire information space as a polygon and the portal position as a polygon superimposed on the larger polygon representing the information space. Alternatively, we could use the message area to indicate the displacement of the user left and/or right of some reference point much as we showed the displacement vertically in previous examples.

The second aspect that we need to worry about is maintaining a semantic connection between the different frames in the portal. In the vertical direction this is not a great problem since as we scan up and down the same part of the template is displayed in the portal. However, in the horizontal direction the part of the template visible in the portal changes as we move left and right. It is therefore desirable to keep some common thread between horizontally adjacent frames. One possibility is to choose one of the template fields as a *key* field and to keep this field in the portal at all times as we move horizontally.

For example, in figure 5 one of the fields of the template, say FILM, could be kept in the portal as we move the portal position horizontally.

7.3 Control Functions

Control functions deal with signalling the system that it should note something about the current state of the system and perform some action based on the state. The effect of invoking a control function can be thought of as invoking a procedure that takes as its input the current system state and/or field value. Depending on the value of the input parameter(s), certain actions occur. Control functions can be provided in one of several ways, e.g., as function buttons or menu commands.

SELECT

The **select** function takes as its input parameter the current field value. The result of the function depends on the characteristics of the field value. If we select a category field value, then the result is to make that category part of the route we are following. Selecting a name in the **ROUTE** field causes the system to back up our routing to that point. As we will see, selecting a data value that is of type *image* or *voice* results in the display of a picture, the showing of a film or videotape, or the playback of recorded voice messages. If a field value has no explicit procedure defined for it, then selecting it has no effect, i.e., a null action results.

MARK

The **mark** function takes as its input parameter the current field value. The result of the function is that the system notes the marked value for future reference. For example, when we are examining field values we can mark certain ones for further action. Marking can only be done in **display** specification mode.

DESCRIBE

The **describe** function takes as its input parameter the current field value. The result of the function is that the system provides a detailed explanation of the current field value. This explanation can be visual, verbal or both. For example, for a category field value the **describe** function can be used to get an explanation of the information available in the category. Not all field values need have descriptions associated with them. Therefore, invoking this function can result in no description, i.e., a null action.

HELP

The **help** function takes as its input parameter the current system state. The result of invoking the **help** function is to receive instructions as to what actions can be performed by the user, and how, given the current system state.

CANCEL

The **cancel** function takes as its input parameter the current system state. The result of the function is to undo the last control function or the current specification mode.

SUSPEND

The **suspend** function takes as its input parameter the current system state. The result of the function is to save the current state of the user interaction. A suspended user interaction can be reactivated at any time. The interaction proceeds from the point of suspension as if it had never been interrupted. Suspended user interactions are kept in the data base named 'suspended' and can be reactivated by selecting them.

EXIT

The **exit** function takes as its input parameter the current system state. The result of the function is to end the current user interaction. No information about the current interaction is saved.

SAVE

The **save** function takes as its input parameters the current system state plus a user supplied name. The result of the function is to save the specification of the user interaction and to name it explicitly. The interaction can subsequently be duplicated by selecting the user defined name. The user defined name becomes part of the structure part information for this user. The **save** function provides a view definition capability. The **save** function differs from the **suspend** function in that a suspended interaction is only maintained temporarily until it is completed. A saved interaction on the other hand is maintained permanently by the system. A saved interaction can be dropped by saving a null (initial) system state with the same name as a currently saved system state.

7.4 Specification Modes

Specification modes deal with the way in which a user specifies requests for information to the system. Invoking a specification mode implies that a sequence of actions will follow to form the specification. A specification mode is ended implicitly by the **select** function, **save** function or another specification mode, or explicitly by the **cancel** function or **exit** function. When interacting with external data bases, there are two ways that the user can specify his request. One of these will be the default mode; which one is determined by the user. The default mode is entered initially when the user signs on and after every **select** function or **save** function.

FILL

Fill mode allows the user to select various fields and to specify a value or values for each field selected. Fields are selected by pointing at them. They are filled in explicitly by entering a value from a keyboard or implicitly by marking displayed values as explained under **display** mode.

If multiple values are specified for a field, then the *or* boolean operation is assumed among the values, i.e., match where value-1 or value-2, etc. Note that an *and* boolean among values does not make sense unless an entry within a field in fact can have multiple values. If more than one field is

filled in, then the *and* boolean operation is assumed among fields, i.e., match where field-1 and field-2, etc. An *or* boolean among fields can be handled by filling multiple templates.

The end of template filling is signalled by entering **display** mode with the current field being the template. The user is then shown all entries of the template that match the specification. If no fields have been filled in, then all entries corresponding to the template are displayed.

DISPLAY

Display mode allows the user to look at entries of fields or templates in the portal. If the template is empty when **display** mode is invoked, then all entries are displayed. If the template has been filled in, then only entries that match the filled in template are displayed as discussed under **fill** specification mode.

In **display** mode the template entries can be scanned using the pointing and portal positioning operations. The current field and entry can be selected using the **select** function which invokes the procedure associated with the current field value. Field values can also be marked in **display** mode. Subsequently, entering **display** mode again will select only marked field values for display. If the current field is the template, then marking has the effect of selecting only the marked entries for display. If the current field is a field within the template, then the effect is to select only those entries that match the marked field values. The same conventions apply in this case as for **fill** mode.

7.5 Data Types

Most traditional query languages handle only formatted data in a reasonable manner. However, there are other types of data that may need to be dealt with when querying external data bases. Some of these are: formatted text data nonformatted text data image data, and voice data.

Because of the nature of the different types of data, the way in which they are queried is specific to the data type. However, the same interaction protocols can be used on all types of data.

FORMATTED TEXT DATA

Once we reach the data part (instance level) in the querying process, we are presented with a template of the format of the data part. An example of such a template is shown in figure 6.

We can select from such a template by using the **fill** and/or **display** specification modes discussed above. The default for field filling assumes that we want an exact match for the value specified. For certain types of fields we can also allow other kinds of matches such as pattern matches (find a certain pattern in a field), less than, less than or equal, greater than, greater than or equal and not equal matches. These options can be indicated by prefixing the specified value with a suitable symbol, e.g., $<$, $\leqslant$, etc. Fields to be filled can be selected using the pointing operation discussed earlier.

NONFORMATTED TEXT DATA

If the data part consists of nonformatted text data, then when we arrive at the data part we are again presented with a template of the nonformatted text data. An example of such a template is shown in figure 7.

We can select from text by specifying a pattern for which we are looking in the text. We do this by filling in the text pattern in the appropriate field. Displaying the specified pattern causes the system to

**** VIDEOTEX INFORMATION SERVICE ****

ROUTE: Books.Novel.Gone with the Wind.Chapter 1

MESSAGE

Fig. 7 Example nonformatted data.

show us the pattern in context, i.e., where it appears in the text. In this case, the *next* and *previous* options for the **entry** operation will show us the next and previous entries with the pattern in context (if the current field is the template). We can also look at other frames around the current frame with the **frame** operation. Entering **display** mode without form filling will show us all the text.

IMAGE AND VOICE DATA

Certain fields within formatted or nonformatted text data may correspond to image or voice data. Currently it is very difficult to query these types of data directly. In the future it may be possible to input a picture or a voice message and ask the system to find all instances of these. However, for now it is only possible to display or play back the image or voice data. Accordingly, this playback or display action can be initiated by selecting the appropriate entry within an image or voice field by the **select** function. In order to indicate to the user the nature of these fields, they can be suitably highlighted by specific colours or marked by special graphic symbols, e.g., an icon of a camera to indicate photographic pictures.

8 CONCLUDING REMARKS

In this report, we examined the process of querying via an interactive query language in general and with respect to external data bases in particular. We first divided the querying process into three parts: request, reply and dynamics. For each part we identified several characteristics and indicated desirable properties of these characteristics. We then identified seven categories of query languages and evaluated each type with respect to these characteristics. From this evaluation, we determined some desirable properties for a query language for external data bases. Finally, we proposed a design for such a query language.

In developing our design for an interactive query language for external data bases we concentrated on the request and dynamics aspects of the query language. For requests the parameters of user interaction were: number of keystokes required, type and number of commands available, degree of freedom in request formulation, selectivity of requests, uniformity of requests with respect to data bases accessed and the ability to customize the request formulation for a specific application. For the dynamics of interaction the parameters related to: the bandwidth between the user and the system, the degree of gamesmanship involved in the interaction, number of protocols available for interaction, the responsiveness of the system to a user request and the degree of control experienced by the user in an interaction. Let us examine the proposed design with respect to these properties.

The number of keystrokes required to specify a request depends on the specification mode chosen by the user. If the user displays and marks entries, then he need only point at, mark and select values. The number of keystrokes required will depend on the pointing mechanism available. Marking requires one keystroke for each value marked while selecting also requires one keystroke for each value selected. If fill mode is used, then the user may have to point at fields and type in field values. However, even in this case many keystrokes can be avoided if the user displays field values and marks them.

In the proposed design there are two specification modes, eight control functions and possibly (depending on the pointing mechanism) one to three pointing operations. One of the two specification modes will be the default mode so the user need not specify it explicitly. Of the eight control functions, two are likely to be used heavily — **select** and **mark**. If these functions are implemented as function buttons and labelled with the function names, then there will be no need for the user to remember the function names. In addition, the functions are named so as to clearly convey their meaning and on-line help is always available via the **help** function.

Because of the way the user interaction has been designed, it is very difficult to formulate a request incorrectly, either syntactically or semantically. Syntactically, the query language has no syntax to remember. The specification modes accept any input state for the templates, from empty to fully filled. Control functions are always valid although they may return a null result. Semantically, field names always appear in the portal and the user can see the structure of the data via the template. It is possible to specify a complicated request incorrectly, e.g., if there are many fields to be qualified with *and*'s and *or*'s. However, the way in which fields on a template interact by default (*and*'s between fields, *or*'s within a field) is fairly intuitive. It must be accepted that complicated requests *are complex* and therefore prone to error in their specification.

The selectivity of the query language is almost as good as keyword and by example query languages. The only difficult operation to specify is a join-like operation (in fact we have not said how to do it). One possibility is to use a saved interaction as input to another interaction. However, we do not feel that there will be a general need for a join-like capability in a query language for external data bases. In any case, one can always have the option of going to a more experienced user interface, i.e., a keyword query language, for such requests.

In the proposed query language, we did not make any assumptions about the structure of the external data bases. The nature of the interface chosen — templates — does not presuppose any particular underlying structure for the data base. Thus, the query language can be used to interface to any type of data base provided suitable translation algorithms are provided to translate the operations and structures in the query language to the operations and structures of the DBMS of the external data base. This is currently an active research area and several approaches to the problem can be taken [Date, 1980; Vassiliou and Lochovsky, 1980].

The query language can be easily customized to particular applications by customizing the templates via which the user interacts with the system. No other aspect of the query language need change to accommodate this customizing. In particular, the hardware required for interaction and the pointing operations, control functions and specification modes remain the same.

The bandwidth of interaction achieved depends on the mode of specification chosen. For display and marking it can be very high. For template filling it can be moderate. Note that the basic concepts behind the interaction remain unchanged as we move from pointing via a keyboard to a mechanical device to voice. Thus, as technology advances allow the bandwidth to increase, the query language can take advantage of these advances.

The gamesmanship of the query language is moderate to good. The user can easily explore alternate paths in the structure with little effort since he can easily save paths and/or back up at any time. This exploration process gradually allows the user to move toward his goal and he can see his progress by the **ROUTE** field in the portal. Depending on how the cross references are constructed, the user can reach the same data from many different paths. The pointing, marking, displaying and filling types of activities involve the user in the interaction. It is not just a matter of pushing a button. The design of the templates and the use of colour and sound can add to the gamesmanship of the query language.

There are several protocols for querying data bases — fill, display and predefined routes. The user can choose the one he feels is most effective for his particular interaction and skill level. He can also set the default specification mode to suit his knowledge level of the system.

Responsiveness may be a limiting factor for the query language with today's technology. **Display** mode especially requires that the system quickly display the requested information. When we are at the data part level this can mean the retrieval of a great deal of information. Being able to display the values of any field implies that there is some fast access mechanism available to quickly retrieve all the values. The use of **display** mode may have to be limited initially to the structure part and certain fields of the data part to overcome the response problems. However, advances in technology may soon make such a restriction unnecessary.

Finally, the entire nature of the user interaction, directing the system to display things, mark things, etc., contributes to the user's feeling of control. He is directing the system to do things for him in reaching his goal. Also, the paradigm for querying introduced earlier helps to contribute to this feeling of control. The paradigm makes the user feel that the system is his friend helping him guide his space ship (the system) through the unfriendly environment of space (the information space) and protecting him from it.

REFERENCES

Bonczek, R.H., Cash, J., and Whinston, A. [1977]. "A Transformational Grammar-based Query Processor for Access Control in a Planning System," *ACM TODS* **2**, pp. 326-338.

Bonczek, R.H., and Whinston, A. [1977]. "A Generalized Mapping Language for Network Data Structures," *Information Systems* **2**, pp. 171-185.

Bown, H.G., O'Brien, C.D., Sawchuk, W., and Storey, J.R. [1978]. *A General Description of Telidon:- A Canadian Proposal for Videotex Systems*, Tech. note 697-E, Communications Research Centre, Department of Communications, Ottawa.

Bown, H., O'Brien, C.D., Sawchuk, W., Storey, J.R., and Treurniet, W.C. [1979]. "Telidon Videotex and User-related Issues," *Proc. Conf. on Visible Languages.*

Burger, J.F. [1977]. "Data Base Semantics in the EUFID System," *Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks*, pp. 202-214.

Carlson, C.R., and Kaplan, R.S. [1976]. "A Generalized Access Path Model and its Application to Relational Data Base Systems," *Proc. ACM SIGMOD*, pp. 143-154.

Chamberlin, D.D., *et al.* [1976]. "SEQUEL2: A Unified Approach to Data Definition, Manipulation and Control," *IBM Journal of Research and Development* **20**, pp. 560-575.

Codd, E.F., Arnold, R.S., Cadiou, J-M., Chang, C.L., and Roussopoulos, N. [1978]. *RENDEZVOUS Version 1: An Experimental English-language Query Formulation System for Casual Users of Relational Data Bases*, Tech. rep. RJ2144, IBM Research Lab., San Jose, Calif.

Date, C.J. [1976]. "An Architecture for High-level Language Database Extensions." *Proc. ACM SIGMOD*, pp. 101-122.

Date, C.J. [1980]. "An Introduction to the Unified Database Language," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 15-32.

Deheneffe, C., and Hennebert, H. [1976]. "NUL: A Navigational User's Language for a Network Structured Data Base," *Proc. ACM SIGMOD*, pp. 135-142.

Denny, G.H. [1977]. *An Introduction to SQL, A Structured Query Language*, Tech. Rep. RA93, IBM Research Lab., San Jose, Calif.

Ellis, C.A., and Nutt, G.J. [1980]. "Office Information Systems and Computer Science," *ACM Computing Surveys* **12**, pp. 27-60.

Gilb, T. and Weinberg, G.M. [1977]. *Humanized Input: Techniques for Reliable Keyed Input.* Winthrop Publishers.

Geudj, R.A., tenHagen, P.J.W., Hopgood, F.R.A., Tucker, H.A., and Duce, D.A. (eds) [1980]. *Methodology of Interaction*, IFIP Workshop on Methodology of Interaction, Seillac, France. North-Holland, Amsterdam.

Hammer, M., Howe, W.G., Kruskal, V.J., and Wladawsky, I. [1977]. "A Very High Level Programming Language for Data Processing Applications," *CACM* **20**, pp. 832-840.

Harris, L.R. [1977] "User Oriented Data Base Query with the ROBOT Natural Language Query System," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 303-311.

Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., and Slocum, J. [1978]. "Developing a Natural Language Interface to Complex Data," *ACM TODS* **3**, pp. 105-147.

Herot, C.F. [1980]. "Spatial Management of Data," *ACM TODS* **5**, pp. 493-513.

Hogg, J. [1981]. *TLA: A System for Automating Form Procedures*, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto.

Hogg, J., Nierstrasz, O.M., and Tsichritzis, D.C. [1981]. "Form Procedures," *in Omega Alpha*, Tsichritzis, D.C. (ed.), Tech. rep. CSRG-127, Computer Systems Research Group, Univ. of Toronto.

Kerschberg, L., Ozkarahan, E.A., and Pacheco, J.E.S. [1976]. "A Synthetic English Query Language for a Relational Associative Processor," *Proc. 2nd Intl. Conf. on Software Engineering*, pp. 505-519.

Lacroix, M., and Pirotte, A. [1977a]. ILL: An English Structured Query Language for Relational Data Bases," *in Architecture and Models in Data Base Management Systems*, (Nijssen, G.M., ed.), pp. 237-260, North-Holland, Amsterdam.

Lacroix, M., and Pirotte, A. [1977b]. "Domain-oriented Relational Languages," *Proc. Intl. Conf. Very Large Data Bases*, pp. 370-378.

Lipson, W., and Lapczak, O. [1976]. *LSL User's Manual*, Tech. note 9, Computer Systems Research Group, Univ. of Toronto.

Luo, D., and Yao, S.B. [1981]. "Form Operations By Example — A Language for Office Information Processing," *Proc. ACM SIGMOD.*

McDonald, N., and Stonebraker, M.R. [1975]. "CUPID - The Friendly Query Language," *Proc. ACM Pacific 75 Regional Conf.*, pp. 127-131.

Mantei, M. [1981]. *Disorientation Behaviour in Person-Computer Interaction*, Ph.D. thesis, Dept. of Communications, Univ. of Southern California (to appear).

Martin, J. [1973]. *Design of Man-Computer Dialogues.* Prentice-Hall, Inc., Englewood Cliffs, N.J.

Mehlmann, M. [1981]. *When People Use Computers — An Approach to Developing an Interface.* Prentice-Hall, Inc., Englewood Cliffs, N.J.

Moorhead, W.G. [1976]. *GXRAM - A Relational Data Base Interface for Graphics*, Tech. rep. RJ1735, IBM Research Lab., San Jose, Calif.

Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., and Wong, H. [1975]. "TORUS - Natural Language Understanding System for Data Management," *Proc. 4th Intl. Joint Conf. Artifical Intelligence*, pp. 414-421.

Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., and Wong, H. [1976]. "TORUS: A Step Toward Bridging the Gap Between Data Bases and the Casual User," *Information Systems* 2, pp. 49-64.

Nierstrasz, O.M. [1981]. *Automatic Coordination and Processing of Electronic Forms in TLA (An Intelligent Office Information System)*, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto.

Prenner, C.J., and Rowe, L.A. [1977]. *A Data Base Oriented Programming Language*, ERL Memo, Electronic Research Laboratory, Univ. Calif. Berkeley.

Reisner, P. [1977]. "Use of Psychological Experimentation as an Aid to Development of a Query Language," *IEEE Trans. on Software Engineering* **3**, pp. 218-229.

Reisner, P. [1981]. "Human Factors Studies of Database Query Languages: A Survey and Assessment," *ACM Computing Surveys* **13**.

Reisner, P., Boyce, R.F., and Chamberlin, D.D. [1975]. "Human Factors Evaluation of Two Data Base Query Languages - SQUARE and SEQUEL," *Proc. AFIPS* **44**, *NCC*, pp. 447-452.

Sagalowicz, D. [1977]. "IDA: An Intelligent Data Access Program," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 293-302.

Schmidt, J.W. [1977]. "Some High Level Language Constructs for Data of Type Relation," *ACM TODS* **2**, pp. 247-261.

Sharman, G.C.H. [1977]. "Update-by-dialogue: An Interactive Approach to Data Base Manipulation," *Proc. ACM SIGMOD*, pp. 21-29.

Shneiderman, B. [1978]. "Improving the Human Factors Aspect of Database Interactions," *ACM TODS* **3**, pp. 417-439.

Shneiderman, B. [1980]. *Software Psychology: Human Factors in Computer and Information Systems.* Prentice-Hall, Inc., Englewood Cliffs, N.J.

Spath, C.R. and Schneider, L.S. [1977]. "A Generalized End-user Facility Architecture for Relational Data Base Systems," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 359-369.

Stonebraker, M.R., and Rowe, L.A. [1977]. "Observations on Data Manipulation Languages and Their Embedding in General Purpose Programming Languages," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 128-143.

Thomas, J.C. [1977]. "Psychological Issues In Data Base Management," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 169-185.

Thomas, J.C., and Gould, J.D. [1975]. "A Psychological Study of Query-by-example," *Proc. AFIPS* **44**, *NCC*, pp. 439-445.

Vandijck, E. [1977]. "Towards a More Familiar Relational Retrieval Language," *Information Systems* **2**, pp. 159-169.

Vassiliou, Y., and Lochovsky, F.H. [1980]. "DBMS Transaction Translation," *Proc. COMPSAC '80*, pp. 89-96.

Woods, W.A. [1973]. "Progress in Natural Language Understanding - An Application to Lunar Geology," *Proc. AFIPS* **42**, *NCC*, pp. 441-450.

Zloof, M.M. [1975a]. "Query-by-example: The Invocation and Definition of Tables and Forms," *Proc. ACM Intl. Conf. Very Large Data Bases*, pp. 1-24.

Zloof, M.M. [1975b]. "Query-by-example," *Proc. AFIPS* **44**, *NCC*, pp. 431-438.

Zloof, M.M. [1977]. "Query-by-example: A Data Base Language," *IBM Systems Journal* **16**, pp. 324-343.

Zloof, M.M. [1980]. *A Language for Office and Business Automation*, IBM Tech. rep. RC8091, Yorktown Heights, New York.

LANGAGES D'INTERROGATION INTERACTIVE

POUR

BASES DE DONNÉES EXTERNES

F.H. Lochovsky

D.C. Tsichritzis

Mars 1983

Groupe de recherche sur les systèmes informatiques
Université de Toronto
Toronto (Ontario) M5S 1A1
Canada

Pour: le ministère des Communications, Ottawa, Canada
En vertu du contrat passé avec le ministère
des Approvisionnements et Services, numéro d'ordre OSU80-00124
Période: du 1er juillet 1980 au 31 mars 1981

Des exemplaires de ce document
sont disponibles gratuitement auprès de la:

Direction générale de l'information
Ministère des Communications
300, rue Slater
Ottawa (Ontario)
K1A 0C8

No. DOC-TBR-DBRE-81-5-F

RÉSUMÉ

La quantité de données informatisées disponibles s'accroît sans cesse. Une bonne partie de ces données sont informationnelles ou destinées à la consultation et elles pourraient éventuellement être mises à la disposition du grand public. Jusqu'à récemment, il n'existait pas de support commode qui permette au public d'avoir accès à cette information. Cependant, deux événements, soit la croissance de réseaux de communication et l'arrivée sur le marché des systèmes vidéotex, offrent maintenant au public la possibilité d'accéder facilement à cette information. Dans le présent rapport, nous examinerons la nature du mécanisme qui permet, depuis le foyer et grâce à un service d'information vidéotex, d'avoir accès à des bases de données externes au système vidéotex proprement dit.

Nous étudierons d'abord le processus d'accès à une base de données (interrogation), d'après les caractéristiques de l'interaction du point de vue de l'utilisateur; (dans les pages qui suivent, nous parlerons de l'"interaction de l'utilisateur"). Nous divisons le processus de l'interaction en trois parties: la demande, la réponse et la dynamique. Pour les trois parties, certaines caractéristiques qui décrivent les paramètres de l'interaction de l'utilisateur, sont identifiées. Dans le cas des demandes, les paramètres sont le nombre de frappes requises, le type et le nombre de commandes disponibles, le degré de liberté quant à la formulation de la demande, la sélectivité des demandes, leur uniformité par rapport aux bases de données auxquelles on a accès et la possibilité de personnaliser la formulation de la demande en fonction d'applications précises. En ce qui a trait aux réponses, les paramètres sont la forme dans laquelle la réponse est présentée à l'utilisateur, la question de savoir si la réponse peut être présentée par l'intermédiaire de plus d'un support, si la réponse peut être personnalisée, le degré de maîtrise dynamique sur la réponse et la possibilité de conserver la réponse et de l'utiliser plus tard comme entrée. Pour ce qui est de la dynamique de l'interaction, les paramètres concernent l'intensité de l'interaction entre l'utilisateur et le système, le degré de ludicité que comporte l'interaction, le nombre de protocoles disponibles pour l'interaction, la souplesse du système lorsque l'utilisateur fait une demande et le degré de commande dont il bénéficie dans une interaction. Pour chacun de ces paramètres, une "valeur" souhaitable est déterminée.

Nous isolons ensuite sept types de langages d'interrogation: le langage à mots-clés, par exemples, naturel, à menus, à graphiques, à supports multiples et à jeux informatiques. Chaque genre de langage d'interrogation est décrit et évalué en fonction des paramètres d'interaction.

Il ressort de cette évaluation des langages d'interrogation qu'aucun genre de langage ne réunit à lui seul toutes les caractéristiques souhaitables. Nous proposons par conséquent certaines exigences auxquelles devrait répondre un langage d'interrogation pour les bases de données externes. Compte tenu de ces exigences, nous proposons ensuite un modèle de langage d'interrogation qui donne accès aux bases de données externes.

Pour décrire un langage d'interrogation, un paradigme représentant un vaisseau spatial personnel naviguant dans l'espace à l'aide de cartes détaillées est utilisé. Essentiellement, l'utilisateur peut demander des cartes de l'espace informationnel parmi celles qui sont à sa disposition et choisir sa destination à partir de ces cartes. Celles-ci sont présentées sous la forme de modèles qui représentent la structure et(ou) le contenu des bases de données externes. La plus grande partie de l'interaction est accomplie en pointant les zones de modèles et en entrant les valeurs ou en demandant l'affichage de ces dernières. La nature de l'interaction permet de manipuler de manière uniforme les données mises en forme, les données d'un texte non mises en forme, les données sous forme d'image et les données sous forme vocale.

Langages d'interrogation interactive pour bases de données externes

TABLE DES MATIÈRES

1 INTRODUCTION 1

2 PARAMÈTRES D'INTERACTION DE L'UTILISATEUR 3

2.1 Demande 5
2.2 Réponse 8
2.3 Dynamique 10
2.4 Résumé 13

3 ÉVALUATION DU LANGAGE D'INTERROGATION 15

3.1 Langages à mots-clés 15
3.2 Langages par exemples 17
3.3 Langage naturel 20
3.4 Menu 22
3.5 Graphiques 24
3.6 Supports multiples 26
3.7 Jeux informatiques 29

4 BESOINS EN MATIÈRE D'INTERROGATION 33

5 PARADIGME 39

6 VUE D'ENSEMBLE DE LA CONCEPTION 43

6.1 Choix d'une base de données externe 43
6.2 Exploration des données 51
6.3 Itinéraires préétablis 53

7 CONSIDÉRATIONS EN MATIÈRE DE CONCEPTION 55

7.1 Pointage 55
7.2 Positionnement dans le portique 57
7.3 Fonctions de commande 60
7.4 Modes de spécification 63
7.5 Types de données 65

8 REMARQUES ET CONCLUSIONS 69

BIBLIOGRAPHIE 75

Langages d'interrogation interactive pour bases de données externes

1 INTRODUCTION

Au cours des dernières années, on a assisté à une croissance marquée du nombre des sources d'information informatisées (bases de données) auxquelles on a accès en direct. Beaucoup de ces bases de données en liaison directe fournissent de l'information au public. C'est le cas, par exemple, des systèmes des bibliothèques, des transporteurs aériens et des services d'information des gouvernements. Très souvent cependant, le public n'a pas d'accès direct et facile à ces bases de données. Un intermédiaire interroge la base de données et transmet la réponse ou encore, l'information est donnée de façon non sélective dans des endroits précis.

Une autre réalisation des dernières années a été l'installation et l'expansion des réseaux de communication. Ces réseaux permettent d'interroger les bases de données à distance et à un coût raisonnable. Il devient ainsi possible au public d'accéder en direct aux bases de données "d'information destinée au consommateur". Mais une question se pose: comment et de quel endroit se fera cet accès?

Ces années ont aussi été marquées par l'arrivée des systèmes vidéotex. Il s'agit de systèmes interactifs de communication visuelle dont l'un des buts est de permettre au public d'accéder à des bases de données externes. (Par base de données externe, on désigne une base de données dont le contenu et la structure ne relèvent pas du système de livraison de l'information (vidéotex).) Télidon est l'un de ces systèmes (Bown et autres, 1978). L'interaction avec le système se fait par l'intermédiaire d'un téléviseur modifié, qui sert de terminal de visualisation, et d'un clavier ou d'un bloc de touches, qui est utilisé pour choisir les données à afficher.

Comme le public aura un jour accès aux systèmes vidéotex, l'interaction de l'utilisateur avec ces systèmes soulève plusieurs questions (Bown et autres, 1979). Comment, par exemple, le public demandera-t-il l'information des diverses bases de données externes qu'il aura à sa disposition? En effet, il existe actuellement un certain nombre de langages, tous différents les uns des autres, ayant chacun son vocabulaire et sa syntaxe propres.

Il semble peu souhaitable, du moins à première vue, d'obliger l'utilisateur d'un système vidéotex à apprendre et retenir un langage d'interrogation distinct pour chaque base de données externe qu'il voudrait consulter. Pour que les utilisateurs acceptent les services vidéotex de recherche de l'information, il ne devrait y avoir qu'une seule manière d'interroger et le public devrait pouvoir l'apprendre et l'utiliser facilement. Avec ce langage d'interrogation, les gens pourraient obtenir l'information de toute base de données externe reliée au réseau de communication.

Dans le présent rapport, nous définissons et décrivons plusieurs catégories de langages d'interrogation interactive. Nous établissons quelques critères d'interaction des usagers qui permettent d'évaluer ces catégories de langages d'interrogation. Nous faisons une brève étude de chaque catégorie en fonction des critères énumérés ci-dessus et, en dernier lieu, nous proposons une méthode particulière d'aborder un langage d'interrogation pour les bases de données externes.

2 PARAMÈTRES D'INTERACTION DE L'UTILISATEUR

L'interaction personne-ordinateur, a fait l'objet de nombreuses études récentes (Martin, 1973; Gilb et Weinberg, 1977; Guedj et autres, 1980; Schneiderman, 1980; Mehlmann, 1981). Cependant, beaucoup moins nombreuses sont celles qui portent sur les langages d'interrogation interactive (Reisner, 1981). En utilisant certains résultats des études qui ont porté respectivement sur l'interaction personne-ordinateur et sur les langages d'interrogation interactive, nous étudierons le processus d'interrogation de la base de données du point de vue de l'utilisateur. On peut diviser le processus en trois parties: la demande, la réponse et la dynamique.

La demande a trait à la formulation de l'interrogation du système par l'utilisateur. Ce dernier informe le système de ce qu'il attend de lui. Ce qui nous intéresse ici, ce sont les demandes d'information en mémoire. On utilise habituellement une forme de **langage** que l'utilisateur et le système acceptent tous deux. Le langage peut prendre plusieurs formes: visuel, oral, gestuel, etc.

La réponse vise à informer l'utilisateur du résultat de l'action. Ce résultat peut être la réponse à la demande ou une information sur le statut d'une demande. Les données de sortie devraient être présentées afin que l'utilisateur puisse facilement les comprendre et les assimiler. Encore là, il faut une forme de langage pour communiquer les données de sortie. La visualisation semble être un choix naturel pour le système vidéotex. Il peut s'agir de tableaux, de graphiques, d'images, etc. La présentation sonore est une autre solution possible.

Enfin, la **dynamique** a trait à la nature de l'interaction entre l'utilisateur et le système. Les processus de demande et de réponse

doivent se faire par un moyen qui permette à l'utilisateur et au système de communiquer l'un avec l'autre, c'est-à-dire que les langages doivent disposer de moyens de transport entre les deux **participants**. Ainsi, l'interaction peut se faire par clavier, oralement, par pointage, etc. Du point de vue de l'utilisateur, la qualité de l'interaction est un aspect très important du processus d'interrogation. Dans un système vidéotex, l'interaction devrait, selon nous, être intéressante et facile pour l'utilisateur.

Dans la plupart des langages d'interrogation, les trois parties mentionnées ci-dessus sont considérées en bloc. C'est là une erreur puisque le langage utilisé pour formuler les demandes n'a pas, pour l'utilisateur, les mêmes exigences qu'un langage utilisé pour donner les réponses. De plus, la dynamique d'un langage est indépendante des aspects statiques de la formulation des demandes et de la présentation des réponses. Par conséquent, on peut étudier les exigences relatives à la dynamique indépendamment des exigences de la demande et de la réponse.

On peut dès lors définir certaines caractéristiques pour chaque partie du processus d'interrogation. Ces caractéristiques décrivent les paramètres de l'interaction des utilisateurs. Les **valeurs** qu'affichent ces paramètres dans un langage d'interrogation donné déterminent si ce langage a, ou n'a pas, les caractéristiques **souhaitables** en ce qui a trait à l'interface personne-ordinateur. Cette section sera consacrée à l'identification et à la définition de plusieurs caractéristiques pour chaque partie du processus d'interrogation. De plus, nous indiquerons ce qui, selon nous et selon les études mentionnées ci-dessus, constitue une valeur **souhaitable** pour chaque caractéristique. La liste des valeurs **souhaitables** contituera alors les exigences d'un langage **souhaitable** pour l'interrogation interactive.

2.1 Demande

Nous identifions ici six caractéristiques de formulation des demandes qui sont importantes pour l'évaluation des langages d'interrogation interactive.

FRAPPES

Les frappes sont la quantification des données introduites par l'utilisateur pour que le système comprenne entièrement la demande de l'utilisateur. Si l'on utilise un clavier pour formuler la demande, la mesure se fera en comptant le nombre de touches frappées. Si la demande est formulée en pointant des articles, c'est le nombre d'articles pointés qui constituera la mesure. Il est souhaitable de **réduire au minimum** le nombre de frappes nécessaires à la formulation de la demande. Ainsi, la formulation sera rapide et facile et l'on évitera les possibilités d'erreurs, les mauvaise touches, par exemple.

COMMANDES

Les commandes sont l'ensemble des instructions dont dispose l'utilisateur pour signifier au système ce qu'il a à faire. Ainsi, l'utilisateur peut demander au système de **rechercher, d'imprimer** ou de **dessiner.** Le nombre des commandes doit être **restreint** pour qu'on puisse les mémoriser facilement. Les commandes doivent être **simples**, c'est-à-dire que la syntaxe d'une commande et le nombre de choses à spécifier pour utiliser la commande doivent être faciles à mémoriser et l'utilisateur doit pouvoir s'en servir sans hésiter.

FORMULATION

La formulation d'une demande concerne la difficulté plus ou moins grande que l'on a à spécifier une demande dont la syntaxe ou la sémantique sont incorrectes. Une demande dont la syntaxe est incorrecte ne suit pas la structure que reconnaît le système. Il s'agit souvent de mots comportant une faute d'orthographe ou encore d'une demande incomplète. Une demande incorrecte sur le plan de la sémantique en est une à laquelle le système ne peut donner de réponse parce qu'il n'a pas l'information requise; cela peut encore être une demande pour laquelle il y a eu une réponse qui n'est pas celle que l'utilisateur attendait, c'est-à-dire, que l'interprétation de la demande par le système diffère de celle de l'utilisateur. Il est souhaitable qu'un langage d'interrogation élimine la possibilité de demandes contenant des erreurs de syntaxe ou de sémantique ou, du moins, qu'il rende difficile la formulation de demandes contenant de telles erreurs.

SÉLECTIVITÉ

La sélectivité est la possibilité qu'a l'utilisateur de spécifier aussi précisément que possible les données qu'il désire retrouver. Si un langage d'interrogation comporte une grande sélectivité, l'utilisateur n'obtiendra que les données voulues. Si, par contre, la sélectivité en est faible, beaucoup des données recouvrées seront du **bruit**, c'est-à-dire qu'elles ne répondront pas à la demande. Il est donc souhaitable que le langage ait une **grande** sélectivité, pour que l'utilisateur ne soit pas distrait par des données parasites. De plus, si la facturation se fait en fonction des données recouvrées, une grande sélectivité peut contribuer à réduire les coûts d'utilisation du système.

UNIFORMITÉ

L'uniformité concerne l'indépendance du langage d'interrogation par rapport au genre de base de données à laquelle il donne accès et l'application pour laquelle le langage est utilisé. Pour des bases de données, qui peuvent être de divers types et structurées de façons différentes, la question de l'uniformité se pose ainsi: le langage d'interrogation peut-il être utilisé pour accéder à toutes ces bases de données? Pour les applications, le langage peut-il être utilisé pour tout genre d'application ou la structure de l'application est-elle spécifique? Ainsi, si l'on peut utiliser un langage d'interrogation pour avoir accès à de l'information sur la bourse, la météo, les nouvelles, l'épicerie, etc., on a un langage uniforme, du point de vue de l'application. Il est souhaitable que la structure d'un langage d'interrogation soit uniforme, et pour les bases de données et pour les applications. Dans la pratique, il est facile d'obtenir l'uniformité d'application, tandis qu'il est beaucoup plus difficile d'obtenir l'uniformité dans le cas des bases de données.

PERSONNALISATION

Quand on parle de personnaliser un langage d'interrogation, cela signifie en adapter la forme à celle d'une application particulière. Cela a pour but de faciliter la formulation des demandes en fonction de l'application, puisque la forme du langage d'interrogation aide l'utilisateur à mémoriser le langage. Ainsi, certaines applications utilisent, pour des opérations données, des mots spécifiques qui pourraient servir de mots-clés dans un langage d'interrogation. De cette façon, la sémantique d'une demande est acheminée jusqu'à l'utilisateur plus facilement et avec plus de précision. Il est souhaitable qu'un langage d'interrogation fasse appel à autant de concepts d'application que possible pour faciliter l'interaction de l'utilisateur avec le système.

2.2 Réponse

Nous identifions ici cinq caractéristiques qui servent à représenter les réponses; ces caractéristiques sont importantes pour l'évaluation des langages d'interrogation interactive.

PRÉSENTATION

La présentation est la forme dans laquelle la réponse est donnée à l'utilisateur. Le facteur le plus important est la complexité de la présentation. Une façon de mesurer cette complexité consiste à mesurer le temps que met l'utilisateur à saisir le contenu de la réponse. Si la réponse est donnée sous la forme d'un dialogue que l'utilisateur doit lire sur un écran de visualisation, il faudra peut-être un certain temps pour comprendre l'information donnée en réponse. Par contre, les graphiques ou les tableaux peuvent ordinairement être interprétés beaucoup plus rapidement. De plus, le même espace permet de présenter beaucoup plus d'information. Il est souhaitable que le langage d'interrogation affiche la réponse à une demande sous une forme que l'utilisateur comprendra et suivra sans difficulté.

PRÉSENTATION SUR SUPPORTS MULTIPLES

Les gens communiquent de plusieurs façons. Par conséquent, s'il est souhaitable que l'utilisateur puisse choisir la forme que prendra une réponse, il est également souhaitable qu'il puisse choisir le support qui servira à lui présenter cette réponse. Ainsi, la réponse pourrait être imprimée sur papier, affichée sous forme de texte, de graphique, ou encore présentée sous forme d'animation sur un écran; elle pourrait même être vocale. De cette façon, les besoins et l'humeur de l'utilisateur détermineraient la formulation de la réponse. Il pourrait vouloir parcourir la réponse à tête reposée, y jeter un coup d'oeil rapide ou encore l'écouter tout en faisant autre chose. Il peut aussi

être important d'offrir le choix entre divers appareils de présentation et divers supports de communication, de façon à permettre aux personnes handicapées d'accéder au système.

PERSONNALISATION

Les langages d'interrogation ne tiennent ordinairement pas compte des personnes auxquelles sont destinées les réponses. Il y a habituellement une seule manière de présenter les réponses, peu importe l'application, le genre d'appareil d'affichage, le moment de la journée et le type d'utilisateur. La personnalisation des réponses, tout comme celle des demandes, signifie que la forme de la réponse est déterminée par les facteurs qui entourent l'interaction. Ainsi, l'affichage de l'information météo se fera mieux graphiquement que par des textes. Les enfants peuvent vouloir obtenir les réponses à leurs questions sous forme de dessins animés qu'ils comprendront mieux que les textes ou les graphiques, plus difficiles à interpréter. Les utilisateurs peuvent choisir des formes qui varieront selon le moment de la journée: en effet, ils préféreront peut-être de courts résumés verbaux tard la nuit tandis qu'ils voudront des textes plus détaillés en début de soirée. Il est souhaitable qu'un langage d'interrogation donne toutes les possibilités de personnaliser les réponses données à l'utilisateur selon les facteurs énumérés ci-dessus.

COMMANDE DYNAMIQUE

L'utilisateur devrait pouvoir régler le rythme auquel la réponse lui est présentée. Cela signifie que, si la réponse est donnée à l'utilisateur sous une certaine forme et en utilisant un certain support, le système doit donner à l'utilisateur un moyen de commander efficacement la présentation. L'utilisateur devrait pouvoir commander la vitesse à laquelle le texte est affiché à l'écran. Il devrait aussi pouvoir commander dynamiquement la vitesse, c'est-à-dire pouvoir revenir

en arrière ou encore sauter des parties de texte. Il devrait aussi pouvoir changer la forme de la réponse, passer du texte à la parole, par exemple, sans devoir formuler sa demande à nouveau. De plus, ces changements ne devraient pas présenter trop de difficultés.

RÉUTILISATION POSSIBLE

Il arrive souvent que la réponse à une demande ne serve pas immédiatement mais qu'elle soit utile plus tard ou puisse servir comme donnée d'entrée pour une autre demande. Il est donc souhaitable que le langage d'interrogation permette de conserver la réponse à une demande pour la traiter par la suite ou pour la réutiliser. De cette façon, il est possible de ramener les interrogations complexes à des fragments logiques plus simples et de réduire ainsi la quantité totale des interactions entre l'utilisateur et le système. Ce mode d'interaction peut se rapprocher davantage de la manière dont les gens s'acquittent de tâches complexes.

2.3 Dynamique

Nous identifions ici cinq caractéristiques d'interaction entre l'utilisateur et le système qui sont importantes pour l'évaluation des langages d'interrogation interactive.

INTENSITÉ

L'intensité de l'interaction est la mesure de la vitesse à laquelle l'utilisateur et le système communiquent entre eux. En général, les gens peuvent absorber plus d'information sous forme d'images que sous forme de texte. L'utilisation d'un clavier pour faire une demande exige beaucoup moins d'intensité que la voix. L'intensité dépend des moyens techniques disponibles pour l'interaction et de la manière dont les demandes et les réponses sont énoncées. Il est souhaitable

d'optimiser l'intensité des communications entre l'utilisateur et le système par l'intermédiaire du langage d'interrogation.

LUDICITÉ

Il est important que l'interaction entre le système et l'utilisateur soit intéressante pour ce dernier. La nature de l'interaction doit donc représenter un défi et le degré de difficulté de l'interaction doit être lié à l'avantage que l'on tire de l'information obtenue. On appelle ce paramètre la ludicité. L'utilisateur du système vidéotex n'a aucune motivation extérieure, par exemple financière, qui puisse le pousser à une interaction avec le système. La motivation est purement intérieure, c'est-à-dire que l'utilisateur veut apprendre quelque chose ou encore se détendre. Il doit sentir que le temps qu'il investit vaut les résultats obtenus, ou du moins que le temps consacré à l'interrogation a été une source de joie et non de frustration. Par conséquent, si la formulation des demandes est intéressante et synonyme de défi, le sentiment d'ennui qu'entraîne l'interaction peut être atténué par sa valeur éducative ou récréative.

PROTOCOLE

Le protocole est la manière spécifique dont l'utilisateur dialogue avec le système. Ainsi, l'utilisateur peut toujours formuler une demande de la même façon en se servant du clavier. Un langage d'interrogation devrait offrir plusieurs protocoles d'interaction entre l'utilisateur et le système. De cette façon, l'utilisateur peut à tout moment choisir le protocole qu'il désire. Il peut être souhaitable de disposer de différents protocoles selon la difficulté des demandes. De cette façon, les demandes simples sont faciles à formuler, un seul bouton à presser par exemple, tandis que les demandes plus complexes exigent plus de travail.

SOUPLESSE

L'idéal serait que le système réponde toujours immédiatement à la demande de l'utilisateur. Cependant, ce n'est pas toujours techniquement possible. À défaut de cette situation idéale, la réponse du système devrait être adaptée à la nature de l'interaction. Par conséquent, si une demande est courte ou très facile à formuler, l'utilisateur peut s'attendre à ce que le système donne rapidement une réponse. Inversement, si la demande est longue ou très difficile à formuler, on peut raisonnablement s'attendre à ce que la réponse tarde à venir. Il est souhaitable que la difficulté à formuler la demande entraîne de la difficulté à la traiter. De cette manière, l'utilisateur n'est pas ennuyé d'avoir à attendre pour obtenir la réponse à une question qu'il trouve très facile à formuler.

MAÎTRISE

Les gens aiment avoir la sensation de contrôler la situation, par eux-mêmes ou par personne interposée. Ils n'aiment pas se sentir menés, et surtout pas par une machine. Quand il dialogue avec un système informatisé, l'utilisateur devrait toujours avoir l'impression qu'il domine la situation et que le système vient l'appuyer. Même si, finalement, c'est le système qui a la maîtrise, il doit susciter chez l'utilisateur le sentiment que c'est plutôt lui qui dirige. La souplesse et la serviabilité du système sont des facteurs déterminants qui contribuent à donner cette sensation. Le degré d'interaction est aussi important. Par exemple, on a davantage l'impression de contrôler la situation quand on est au volant d'une automobile plutôt que dans un autobus en tant que passager passif. Par conséquent, à la suite d'une demande, la réponse du système contribue à donner l'impression que l'on domine la situation. Cette impression peut être le facteur déterminant qui incitera l'utilisateur à accepter le système.

2.4 Résumé

Nous avons exposé à grands traits plusieurs des caractéristiques souhaitables d'un langage d'interrogation. Il arrive quelquefois que les exigences relatives aux caractéristiques souhaitables s'harmonisent bien. Ainsi, l'intensité de l'interaction peut être optimisée en diminuant le nombre de frappes par unité de temps. Dans d'autres cas, ces exigences s'opposent les unes aux autres. Ainsi, plusieurs protocoles dans un langage d'interrogation peuvent nuire à l'uniformité, puisque des protocoles différents peuvent faire appel à des commandes différentes. Il s'agit donc, au moment de concevoir un langage d'interrogation, de définir et d'évaluer les compromis que l'on doit faire entre les diverses exigences pour en arriver à un choix acceptable de paramètres d'interaction pour l'utilisateur. Plutôt que de choisir une valeur, il peut être souhaitable de donner plusieurs versions d'un langage d'interrogation dans lequel on met l'accent sur différentes exigences. L'évaluation des facteurs humains peut aider à décider des meilleures approches et à choisir la valeur des paramètres.

Pour montrer comment on évalue l'interaction de l'utilisateur par rapport à ces exigences, on peut étudier une demande portant sur les prix à la bourse ou sur les rapports météorologiques. Une façon de fournir ce service consiste à avoir un rapport complet sur la bourse ou la météo disponible sur demande. La personne presse un bouton du logiciel identifié **bourse** ou **météo** et le système lui fournit, sous la forme d'un texte suivi, un rapport normalisé sur la bourse ou la météo. Jetons maintenant un coup d'oeil sur les paramètres de l'interaction de l'utilisateur pour évaluer dans quelle mesure ce service est souhaitable.

Pour ce qui est de la formulation de la demande, le nombre de frappes est minimal, la commande est simple, les mauvaises demandes ne peuvent pas être formulées et ce genre de commande peut convenir à

beaucoup d'applications (en utilisant des boutons différents) et peut servir à de nombreuses applications. Cependant, la commande n'a absolument aucune sélectivité. L'utilisateur aura toujours un rapport complet sur la bourse ou la météo. Il ne peut pas isoler un secteur d'affaires, une compagnie ou un moment choisi, pour ce qui est de la bourse, ni une région géographique ou un moment particulier pour ce qui est du rapport météorologique. Il obtient tout ce qu'il y a en mémoire et il a la tâche d'en extraire les données de qui l'intéressent.

Quant à la réponse, la présentation est assez simple et il lui suffit de la faire se dérouler à la vitesse qui lui convient et d'habitude il peut conserver la réponse pour consultation ultérieure. Cependant, l'utilisateur n'a pas souvent le choix du moyen ni du format de présentation de l'information, et ne peut pas non plus adapter la réponse aux possibilités particulières de son terminal, celle de produire des graphiques, par exemple. Il obtient la réponse sous forme de texte parce que cela convient à tous les appareils de sortie. Cependant, la météo peut prendre différentes formes selon que l'utilisateur est un enfant, qui préfère des dessins animés, ou bien un homme d'affaires qui aura besoin d'un rapport précis et concis.

Passons maintenant à la dynamique de l'interaction. Il n'y en a pratiquement pas. La demande est formulée en pressant un bouton. L'intensité est alors très grande du fait de la rapidité avec laquelle l'utilisateur communique la demande au système, mais, par la suite, l'interaction est terminée pour ce qui est de l'utilisateur. Il ne peut intervenir sur la réponse: il est complètement passif. Ce sont là des conditions stériles et ennuyantes pour choisir de l'information.

3 ÉVALUATION DU LANGAGE D'INTERROGATION

Dans la présente section, nous répartissons les langages d'interrogation en sept catégories. Pour chacune d'elles, nous décrirons brièvement la forme des langages. Par la suite, nous parlerons du genre de langage par rapport aux paramètres d'interaction de l'utilisateur décrits à la section 2.

3.1 Langages à mots-clés

Les langages d'interrogation à mots-clés permettent à l'utilisateur de se servir d'une forme restreinte de langue courante pour dialoguer avec le système. Il existe de nombreux exemples de ces genres de langages dont le SQL (Denny, 1977), le ILL (Lacroix et Pirotte, 1977 a, b), et le NUL (Deheneffe et Hennebert, 1976). Les demandes sont formulées en utilisant une grammaire restreinte semblable à celle de l'anglais dont des mots précis (mots-clés) signalent le début d'éléments spécifiques d'une demande. Pour faciliter le traitement de ces langages, la syntaxe des demandes est très rigide. Les éléments doivent être utilisés dans un ordre et un format fixes. Les réponses aux demandes sont présentées à l'utilisateur sous forme de tableaux.

Pour formuler une demande, l'utilisateur doit faire appel à un bon nombre de frappes. Cependant, une fois que la demande a été formulée, elle est très facile à lire et à interpréter puisque la structure de ces langages est étroitement liée à la structure de la phrase anglaise. D'habitude, il n'y a que très peu de commandes, mais elles sont assez complexes et offrent de nombreuses possibilités quant aux exigences précises de l'utilisateur. Par conséquent, il faut un certain temps pour maîtriser le langage en entier, quoique l'on puisse en apprendre des sous-ensembles assez rapidement (Reisner et autres, 1975; Reisner, 1977). Comme chaque commande est complexe et le nombre de frappes élevé, le risque de commettre des erreurs de syntaxe dans les

demandes est élevé. Cependant, ces erreurs sont immédiatement repérées par le système. Il est possible de formuler des demandes sans signification, du point de vue de la sémantique, mais cela devient peu probable si la demande ne contient pas d'erreurs de syntaxe et que la base de données est bien conçue. Les langages d'interrogation à base de mots-clés ont une très grande sélectivité. En effet, l'utilisateur peut spécifier très précisément les données qu'il veut obtenir. Ces langages ne tiennent pas compte de la nature de la base de données, quant à son application, bien qu'ils supposent une certaine structure globale des données, qui peut être relationnelle, par exemple. Donc, ces langages ne sont pas liés à l'application; cependant, les bases de données doivent être d'un genre bien précis. Les mots-clés d'un langage peuvent être personnalisés en fonction d'applications particulières. Les autres aspects de ces langages sont rigides.

Les réponses aux questions sont habituellement présentées sous forme de tableaux. L'utilisateur n'a aucune possibilité de changer la forme de la sortie. Bien que les tableaux soient assez faciles à saisir pour certains genres de sortie, ils peuvent être très difficiles à interpréter s'il existe des relations complexes entre les tableaux. Les données de sortie sont normalement présentées sur écran ou sur imprimante seulement. Ces langages ne permettent pas de personnaliser la sortie pour l'utilisateur. Dans la plupart des cas, les données de sortie sont présentées avec des titres de tableaux et de colonne que l'utilisateur peut modifier à sa guise. L'utilisateur n'a aucune commande dynamique sur la sortie à l'exception de celle que fournit la nature même de l'appareil d'affichage, comme le défilement, par exemple. La plupart des langages à base de mots-clés permettent à l'utilisateur de conserver les résultats d'une demande pour les utiliser par la suite.

La dynamique de l'interaction est très minime dans le cas des langages à base de mots-clé. L'intensité de l'interaction est très

faible. Les demandes doivent être introduites en utilisant un clavier et les réponses, lues sur un écran ou une imprimante. L'interaction ne comporte pratiquement aucune ludicité: tout est décidé d'avance. Il n'existe qu'un protocole pour communiquer avec le système: introduire les demandes avec un clavier dans la forme prescrite par le langage. Dans quelques langages d'interrogation à base de mots-clés, les demandes qui sont difficiles à traiter, comme les sommes logiques dans les systèmes rationnels, sont très simples à préciser. Par conséquent, la souplesse du système ne correspond pas à la complexité de la formulation de la demande. Comme les demandes ne peuvent être formulées que de façon restreinte, le système ne peut pas donner beaucoup de rétroaction avant que l'ensemble de la demande n'ait été précisé. Dans de telles conditions, l'utilisateur n'a pas vraiment l'impression de contrôler le système.

3.2 Langages par exemples

Dans les langages d'interrogation **par exemples** l'utilisateur élabore sa demande en donnant au système un exemple de réponses à sa demande. L'exemple est construit selon le format utilisé par le système pour fournir les réponses. Ainsi, dans le rapport **Interrogation par exemples** (Zloof, 1975a; Zloof, 1977, 1980) des modèles de tableaux sont affichés sur un écran et l'utilisateur remplit le tableau selon les colonnes qu'il désire voir dans sa réponse et selon les valeurs qu'il désire dans chaque colonne. Pour le TLA, des modèles de formules sont remplis pour préciser une interrogation (Hogg, 1981; Hogg et autres, 1981; Nierstrasz, 1981). D'autres chercheurs ont proposé des langages semblables (Hammer et autres, 1977; Luo et Yao, 1981).

Dans ces genres de langage d'interrogation, bien que l'utilisateur ait à faire une certaine quantité d'entrées, le nombre de frappes est relativement limité. Le système fournit beaucoup de données d'entrée comme les noms d'attributs qui seraient nécessaires pour les langages à

mots-clés. L'utilisateur n'a qu'à indiquer (normalement par une seule frappe) les valeurs d'attribut qu'il désire dans la réponse et toute condition qui détermine le choix des valeurs. Les commandes qui indiquent l'opération désirée sont généralement limitées à des codes à un seul caractère. Les commandes de ces langages sont simples et peu nombreuses. Cependant, elles ne se prêtent pas bien à la formulation de demandes complexes qui s'étendent sur plusieurs modèles. On sacrifie quelque peu la lisibilité au profit de la simplicité de l'expression. Il est très difficile de formuler des demandes contenant des erreurs de syntaxe, étant donné la rigidité de la spécification. C'est pourquoi il est difficile de formuler des demandes comportant des erreurs de sémantique. En effet, l'utilisateur est toujours au courant de l'information que possède le système puisqu'elle lui est présentée dans les modèles et seules les demandes concernant cette information peuvent être formulées. La sélectivité de ces langages d'interrogation est comparable à celle des langages à mots-clés. La manière dont les demandes sont précisées ne dépend pas de l'application: seuls les modèles changent, et non la manière dont on les remplit. En autant que les bases de données auxquelles on a accès peuvent être représentées sous la forme des modèles utilisés par le langage, on peut dire que le langage n'est pas lié lui non plus à la base de données. La personnalisation de ces langages en fonction d'une application ne peut concerner que les noms d'attributs à l'intérieur d'une application. Comme la forme du langage est rigide, il est difficile de tirer profit de toute caractéristique spéciale d'une application.

Les réponses à des demandes se présentent à l'utilisateur sous la forme de modèles qui servent à formuler les demandes. Par conséquent, il y a correspondance directe entre les demandes et les réponses. De plus, comme les demandes prennent habituellement la forme de tableaux ou de formulaires, on peut les suivre et les comprendre assez facilement. Cependant, cette caractéristique restreint aussi la réponse du fait

qu'elle est présentée sur un écran ou sur une imprimante. Il est aussi difficile de personnaliser les réponses que les demandes. En effet, la forme des réponses est dictée par celle des demandes. Étant donné la nature des réponses, la seule commande dynamique possible est le réglage du défilement sur l'écran. Cependant, ces langages permettent ordinairement à l'utilisateur de conserver les résultats des demandes pour utilisation ultérieure.

Pour ce qui est de la dynamique, nous nous trouvons dans des conditions de demande/réponse pour lesquelles les deux processus sont disjoints. L'intensité est assez grande pour ce qui est du temps consacré à préciser la demande et elle est liée à la complexité de la demande. Ainsi, une demande simple peut être précisée rapidement, tandis qu'une demande plus complexe exige plus de travail. L'intensité de sortie est limitée par la vitesse à laquelle l'utilisateur peut lire ce que lui présente l'écran. Ces genres de langages ont une ludicité un peu plus grande que les langages à mots-clés. Le travail de spécification de la demande est moins ennuyeux et il y a même des systèmes qui guident l'utilisateur quand il remplit les tableaux ou les formules, en ce sens que le curseur passe directement à l'attribut suivant sans que l'utilisateur n'ait à provoquer son mouvement. Par contre, il doit toujours spécifier sa demande en remplissant des modèles. En ce qui a trait à la souplesse, nous devons rappeler que certaines demandes, en particulier celles pour lesquelles il faut passer d'un modèle à un autre, présentent quelques difficultés de formulation. Les demandes difficiles à formuler sont aussi difficiles à traiter. Étant donné le contexte demande/réponse, l'utilisateur n'a pas un contrôle très précis du système. Bien sûr, le système peut donner quelques directives mais sa contribution est restreinte.

3.3 Langage naturel

Les langages d'interrogation basés sur le langage naturel permettent à l'utilisateur anglophone de préciser sa demande dans sa langue maternelle. Le but visé n'est pas de restreindre le vocabulaire de quelque façon que ce soit, bien que, dans la pratique, le système ne puisse traiter qu'une partie d'une langue. Il y a beaucoup d'exemples de ces systèmes qui ont été mis au point par des chercheurs oeuvrant dans le domaine de l'intelligence artificielle (Woods, 1973; Mylopoulos et autres, 1975, 1976; Harris, 1977; Hendrix et autres, 1978). Le système RENDEZ-VOUS mis au point par le laboratoire de recherche IBM à San Jose est peut-être l'exemple le plus connu d'un système qui utilise la technique de la base de données (Codd et autres, 1978).

L'utilisateur formule ses demandes en dialoguant avec le système. Cela signifie qu'il doit utiliser un très grand nombre de frappes pour préciser sa demande. Il n'y a pas de commandes fixes comme telles. Comme c'est le cas pour les phrases énoncées en anglais, les demandes simples sont assez faciles à préciser, et les demandes plus complexes, plus difficiles. Comme l'utilisateur a pleine liberté d'utiliser toutes les ressources de la langue anglaise, il lui est possible de formuler sa demande de plusieurs façons. Cela implique qu'il risque de faire plusieurs genres d'erreurs augmentant ainsi les risques d'une mauvaise interprétation (Schneiderman, 1978). Pour ce qui est de la syntaxe, ces systèmes possèdent dans bien des cas un mécanisme qui permet de corriger les fautes d'orthographe. Ces langages peuvent être très sélectifs, mais cela dépend de la précision de l'utilisateur lorsqu'il exprime sa demande. D'ailleurs, la spécification de la demande peut poser des difficultés puisque le langage ne donne aucune directive à cet égard. Le langage naturel est le plus uniforme que l'on puisse utiliser pour préciser une demande. En effet, il est uniforme pour les applications comme pour les bases de données. De plus, le langage peut être personnalisé par l'utilisateur en fonction de l'application; pour cela, il n'a qu'à faire appel à ses connaissances sur l'application. Cela

suppose que le système possède quelques connaissances sur l'application lui permettant de comprendre le vocabulaire de l'utilisateur.

Les réponses aux demandes peuvent être présentées sous forme de phrases anglaises pour des faits uniques. Pour des faits multiples, il pourrait se présenter des difficultés. Les réponses sont souvent présentées à l'écran sous forme de phrases. Cependant, il est possible d'adapter les phrases à un appareil de sortie vocale. La forme de sortie, tout comme les phrases anglaises, se prête à la personnalisation. Il est possible d'utiliser des mots qui appartiennent à une application ou à un type d'utilisateurs. Parce que l'interaction se fait sous forme de dialogue, l'utilisateur peut exercer une commande dynamique sur la réponse à sa question. L'utilisateur peut communiquer avec le système jusqu'à ce que la réponse contienne l'information qui l'intéresse. Il est difficile de conserver ou de réutiliser la réponse à moins que le système soit capable d'accepter tout nouveau fait qui lui est fourni sous forme de phrases en langage naturel.

La dynamique de l'interaction consiste en un dialogue entre l'utilisateur et le système. Pour les entrées faites par l'intermédiaire du clavier, l'intensité est très faible puisqu'il faut composer des phrases entières en utilisant le clavier. Si la sortie se fait sous la forme de phrases écrites à la machine, l'intensité est aussi très faible. Comme l'utilisateur dialogue avec le système, ce dernier peut introduire une certaine ludicité dans l'interaction, selon la façon employée pour répondre aux demandes de l'utilisateur. Ainsi, il peut demander à l'utilisateur de clarifier une partie de sa demande qu'il ne comprend pas. Il peut reformuler en d'autres termes la demande de l'utilisateur. Cela accorde aussi à l'utilisateur la possibilité de communiquer avec le système de plusieurs façons, par exemple énoncer des phrases, répondre aux questions du système ou choisir entre des possibilités que lui propose le système. Comme les demandes sont formulées par éléments, l'utilisateur a toujours l'illusion qu'il

progresse vers la réponse qu'il cherche. Ce processus permet aussi au système de traiter des demandes complexes en éléments séparés, ce qui a pour effet de réduire les délais entre la formulation de la demande et la réponse. Le processus de formulation de la demande par éléments donne à l'utilisateur l'impression qu'il exerce un contrôle sur le système qu'il guide vers la réponse.

3.4 Menu

Les langages d'interrogation basés sur le menu permettent à l'utilisateur de préciser sa demande en pointant l'action ou les données désirées. À tout moment, l'utilisateur peut choisir dans le menu des données ou des opérations possibles. Le choix d'une opération ou d'un objet peut amener la présentation d'autres menus à l'utilisateur. De cette façon, les demandes sont précisées par étapes selon une structure hiérarchique. On a souvent utilisé ce genre de systèmes et pour beaucoup d'applications. Le système Officetalk mis au point par le centre de recherche de Xerox à Palo Alto, est un exemple d'un système qui interroge les bases de données à l'aide de menus. (Ellis et Nutt, 1980).

Les demandes sont précisées en pointant les actions désirées. Par conséquent, il y a très peu de frappes à utiliser. Les commandes sont simples et elles sont représentées soit par des mots-clé anglais, soit par des icônes qui représentent les actions ou les objets désirés. Ainsi, pour mettre en mémoire un document qui se trouve sur l'écran, on n'a qu'à pointer le document et l'image d'un classeur. Étant donné la structure hiérarchique des menus, les commandes donnent un résultat satisfaisant quand les demandes sont simples et ont une structure hiérarchique, mais elles posent des difficultés dans le cas de demandes plus complexes. La nature très hiérarchisée de l'interaction rend improbable la formulation de demandes incorrectes. C'est le système qui dirige l'interaction à toutes les étapes. La sélectivité est très bonne

pour les demandes qui étaient prévues, c'est-à-dire qui suivent la structure du menu, mais elle est mauvaise pour les autres demandes. On peut utiliser la même technique de choix au menu, peu importe l'application et le genre de base de données. Il est possible de personnaliser les menus selon des applications spécifiques en changeant les mots-clés ou les icônes utilisés.

Les réponses aux demandes sont présentées à l'utilisateur sous forme d'objets stockés dans le système. Dans la plupart des systèmes qui font appel à la technique du menu, il y a des formules qui contiennent un texte structuré ou un tableau. Par conséquent, les réponses présentées de cette manière sont plutôt faciles à suivre. La sortie est disponible seulement sur un écran ou une imprimante et l'utilisateur doit y lire la réponse. Si le système permet de définir de nouveaux objets "au vol", comme c'est le cas avec l'Officetalk, on peut alors personnaliser la sortie pour l'utilisateur. Cependant, cette personnalisation ne porte que sur les genres d'objets conservés par le système, comme des formules, par exemple. L'utilisateur peut commander dynamiquement la présentation de la sortie s'il lui est possible d'utiliser les opérations de menu pendant la présentation de la sortie. Par conséquent, si la réponse peut prendre plusieurs formes, il peut alors les regarder à tour de rôle et les soumettre à une action avant de passer à la suivante. Il se peut que des dispositifs puissent conserver les données de la réponse et les réutiliser par la suite.

La dynamique de l'interaction, pour ces genres de langages, est quelque peu meilleure que pour ceux mentionnés ci-dessus. L'intensité de l'entrée est relativement bonne une fois que l'on a maîtrisé le mécanisme des opérations de pointage. Cette intensité peut être améliorée si l'on utilise des écrans qui permettent l'utilisation de plusieurs fenêtres. L'intensité de sortie est un peu restreinte parce

que l'utilisateur doit lire la réponse. L'interaction par pointage présente des possibilités intéressantes de ludicité. Elle est certainement moins ennuyante que l'utilisation du clavier, et l'emploi d'icônes pour représenter les actions peut contribuer à maintenir l'intérêt de l'utilisateur. Dans le cas du système Officetalk, l'utilisateur peut communiquer avec le système de diverses manières, par exemple il peut passer à un éditeur pour mettre en forme l'entrée et la sortie, ou il peut passer en mode de dessin pour faire des dessins à main levée. Étant donné que l'opération de pointage de la part de l'utilisateur a une grande intensité, l'utilisateur peut s'attendre à ce que toutes les opérations soient d'une même complexité. Cependant, ce n'est pas normalement le cas. Par conséquent, la vitesse et le temps de réponse peuvent ne pas être uniformes, ce qui peut ennuyer l'utilisateur. La nature de l'interaction et sa grande intensité donnent à l'utilisateur l'impression qu'il a la maîtrise du système, qu'il le dirige toujours. Cependant, cela peut ne pas être le cas pour des demandes qui ne suivent pas la structure des menus.

3.5 Graphiques

Les langages d'interrogation à base de graphiques permettent de formuler les demandes surtout sous forme de graphiques, mais il faut habituellement y ajouter quelques données à l'aide du clavier. Chaque genre d'objet de la base de données se voit attribuer une forme géométrique précise. L'utilisateur manipule alors ces formes pour spécifier ses demandes. On utilise des conventions spéciales pour préciser le choix de données spécifiques ou de relations entre les données. Le langage graphique LSL (Lipson et Lapczak, 1976) et le CUPID du système INGRES (McDonald et Stonebraker, 1975) constituent deux exemples de ce genre de langages.

Bien qu'il ne faille qu'un nombre minimal de frappes pour formuler une demande, l'utilisateur doit fournir une bonne quantité de travail

pour construire les diagrammes qui représenteront sa demande. Les objets géométriques doivent être manipulés sur l'écran et les options de traitement spécifiées pour certains d'entre eux. Une fois que la forme de la demande a été spécifiée par un diagramme, il est assez facile de demander l'action nécessaire. Par conséquent, les commandes sont simples; la difficulté réside dans la construction des diagrammes. Il est difficile de formuler des demandes contenant des erreurs de syntaxe puisque c'est le système qui dirige la manière dont les objets sont rassemblés et raccordés les uns aux autres. De même, il est difficile de faire des erreurs de sémantique, quoique cela soit toujours possible. Les langages d'interrogation à base de graphiques peuvent avoir le même degré de sélectivité que les langages à mots-clés puisqu'ils utilisent le même genre de constructions: celles-ci sont tout simplement spécifiées d'une autre manière. Ces langages peuvent servir pour toutes les applications qui utilisent un même genre de base de données, puisque les bases de données utilisent les mêmes genres d'objets pour représenter les données. Pour ce qui est des bases de données de types différents, ces genres de langages peuvent ne pas s'appliquer à moins que l'on spécifie des conventions qui concernent l'interprétation des objets géométriques dans différents contextes. La personnalisation en fonction des applications peut se faire pour les noms d'objets de données employés et les mots-clés utilisés dans le langage.

Les réponses aux demandes sont habituellement présentées sous forme de tableaux, tout comme pour les langages d'interrogation à mots-clés. Par conséquent, les réponses ont à peu près les mêmes caractéristiques en ce qui a trait à la présentation, aux supports multiples, à la personnalisation, à la commande dynamique et à la possibilité de réutilisation.

Dans le cas des langages d'interrogation par graphiques, la dynamique d'interaction n'est pas particulièrement bonne. Les images ont une grande intensité comme moyen de communication, mais on n'obtient pas une très grande intensité quand on les utilise pour formuler des demandes. Il faut beaucoup de temps et de travail pour formuler des demandes par diagrammes. Il peut être plus facile de spécifier la demande par d'autres mécanismes et d'utiliser les diagrammes pour afficher les spécifications, c'est-à-dire que c'est le système qui construit les diagrammes et non l'utilisateur. Les diagrammes sembleraient comporter un haut niveau de ludicité. Il est amusant de construire des diagrammes, mais seulement quand c'est facile de le faire. Ce n'est pas le cas actuellement, pour ces langages, particulièrement dans le cas des demandes complexes. De plus, les utilisateurs inexpérimentés peuvent ne pas savoir exactement comment formuler une demande sous forme de diagrammes. La formulation de demandes, même simples, par des graphiques, peut exiger beaucoup de temps. Par conséquent, l'utilisateur ne peut pas, d'après la formulation de la demande, juger de la complexité de celle-ci. De même, en ce qui a trait à la perception, le processus de spécification de la demande est lent et apparaîtra ainsi à l'utilisateur, étant donné l'interaction restreinte avec le système. On peut douter que l'utilisateur ait l'impression d'avoir la maîtrise de l'interaction.

3.6 Supports multiples

Les langages d'interrogation à supports multiples, comme leur nom le laisse entendre, permettent de communiquer avec le système de plusieurs manières. La formulation des demandes et la présentation des réponses se font sous forme de textes, de données mises en forme par des graphiques, de la voix, de la couleur, de l'animation, etc. Des icônes représentent les catégories de données qui se trouvent dans la base de données. Cette répartition en catégories peut être spécifiée selon

plusieurs niveaux d'abstraction. L'utilisateur peut parcourir les icônes à un niveau particulier et en choisir un en vue d'un traitement ultérieur. En choisissant un icône, l'utilisateur obtient un gros plan des données correspondantes, et soit d'autres icônes qui précisent la catégorie des données, soit les données elles-mêmes. Les icônes peuvent représenter des données textuelles, mises en forme, visuelles ou autres. Les travaux les plus avancés dans ce genre sont ceux du système de gestion des données spatiales qui a été mis au point par le MIT, et le CCA à Boston (Herot, 1980).

Il est possible de formuler les demandes de bien des manières; on peut pointer des icônes, entrer des énoncés par clavier, choisir des menus, etc. L'utilisateur peut choisir les mêmes données de différentes façons, selon son humeur ou ses connaissances de la base de données. Les commandes peuvent être aussi simples que le pointage ou aussi complexes que la spécification du langage d'interrogation à mots-clés. Étant donné la souplesse du processus de formulation des demandes, il est possible de faire fausse route dans la recherche des données. Cela est particulièrement vrai quand l'utilisateur ne fait que parcourir ce qu'il a sous les yeux. Le degré de sélectivité possible dépend en grande partie de la structure de la base de données. Ainsi, s'il y a beaucoup de niveaux d'abstraction entre le plus haut niveau et les données elles-mêmes, le processus de **rapprochement** peut alors être passablement sélectif; sinon, il se peut que l'on choisisse beaucoup de données non pertinentes. Bien sûr, l'utilisateur a toujours la possibilité de choisir un langage d'interrogation plus sélectif. Ces techniques peuvent être utilisées pour toutes les applications et tous les genres de bases de données. Cependant, la technique utilisée, comme le pointage par exemple, peut être plus appropriée que d'autres dans certaines situations. La personnalisation en fonction des applications se fait selon le genre de données stockées pour une application et selon la manière dont elles sont stockées, à chaque niveau d'abstraction d'une base de données.

Les réponses aux demandes dépendent du niveau d'abstraction auquel on se trouve ainsi que de la nature de la base de données. Ces réponses peuvent se présenter sous la forme de textes, d'images, de graphiques ou de toute autre manière, selon ce qui convient à la représentation des données. Cela permet de stocker les données sous une forme qui convienne à l'utilisateur à qui on les destine. Celui-ci peut diriger le système de façon à ce qu'il affiche la réponse d'une certaine manière ou moyennant un certain support. L'utilisateur a la maîtrise entière sur la sortie, sur son débit, sur la forme et sur l'endroit de cette sortie. Bien que les données de sortie puissent être conservées, il n'est pas toujours possible de les réutiliser comme entrée pour une autre demande, si par exemple, elles sont sous forme de graphiques ou sous forme verbale.

La dynamique de l'interaction, est un excellent système. L'intensité de l'entrée et celle de la sortie sont grandes et l'utilisateur peut choisir la forme qui correspond à ses goûts. L'interaction a un degré élevé de ludicité. L'utilisateur peut **survoler** les données et obtenir très facilement un **gros plan** d'une partie en particulier. L'interaction avec le système peut se faire de plusieurs manières et être modifiée au moment de la formulation d'une demande. L'utilisateur peut répondre très rapidement à une question posée par le système. Cependant, il se peut que le système mette beaucoup de temps à répondre à une demande de l'utilisateur étant donné l'interaction de techniques différentes dans l'interface, du moins à l'heure actuelle. Par conséquent, l'interaction peut être source de frustrations pour l'utilisateur, particulièrement si le temps de réponse n'est pas uniforme pour des demandes similaires. L'utilisateur peut avoir une impression très nette de maîtrise dans un tel système. En effet, il existe bien des façons de diriger le système et de lui donner des instructions. Le système peut aider à guider l'interaction de l'utilisateur en surveillant la nature de cette interaction. Ainsi, il peut aider davantage l'utilisateur qui fait beaucoup d'erreurs ou encore

fournir moins de niveaux d'abstraction à l'utilisateur qui emprunte une voie qui lui est familière.

3.7 Jeux informatiques

Beaucoup de jeux informatiques ont été mis au point pour permettre l'accès à de l'information utile ou pour amuser les chercheurs. La plupart de ces jeux ne sont pas connus du grand public car ils utilisent des bases de données très spécialisées qui se trouvent dans des endroits très précis. Les laboratoires Bell, par exemple, ont mis au point un jeu que l'on nomme "EATS" et que l'on utilise pour trouver un restaurant d'un certain genre dans un endroit donné. Le principe de ce jeu consiste à chercher quelque chose en dialoguant avec le système. Pour chaque donnée que vous fournissez au système, celui-ci vous dit quelque chose sur votre situation actuelle. À partir de ce renseignement, vous pouvez dire si vous vous rapprochez ou non de votre but. Pour ces jeux, les techniques d'interaction reposent sur des questions posées par le système ou sur la présentation de choix d'actions par le système à l'intention de l'utilisateur.

Dans un tel contexte, les demandes sont spécifiées progressivement. Ainsi, dans un jeu du genre **question-réponse**, le système pose une question, l'utilisateur donne une réponse, puis le système renseigne l'utilisateur sur l'étape où en est rendu le jeu. Ce genre de formulation des demandes peut comporter l'utilisation de nombreuses frappes, mais les réponses sont habituellement courtes et peuvent n'exiger qu'un oui ou un non. Les commandes sont très simples et, une fois qu'elles sont données, le système guide l'utilisateur afin d'éviter des erreurs de syntaxe dans la demande. Cependant, on peut s'attirer des difficultés sur le plan de la sémantique, car les réponses aux questions sont données dans un contexte limité d'information sur les conditions d'utilisation du moment. Par conséquent, on peut facilement prendre une mauvaise direction. La sélectivité est bonne si l'on suit

une voie directe vers l'information désirée. Cependant, en cas de déviation, on peut obtenir des tas d'informations utiles. Malheureusement, la nature de la formulation de la demande dépend énormément de l'application. Par contre, les techniques utilisées peuvent être très insensibles au genre de base de données sous-jacente. L'interface peut facilement être personnalisée en fonction des applications particulières.

Les réponses sont présentées à l'utilisateur sous forme d'éléments. Une partie de la réponse peut lui être présentée ou il doit la découvrir à mesure que se fait la formulation de la demande. Par conséquent, il est normalement très facile de suivre la présentation puisque l'on ne reçoit que quelques faits à la fois. Les réponses sont normalement présentées sur un écran, mais il est possible d'utiliser des effets sonores pour indiquer que l'on s'approche du but ou que l'on est sur la bonne voie, comme c'est le cas pour les jeux vidéo. L'utilisateur peut régler le débit de sortie de la réponse en contrôlant la vitesse à laquelle il répond au système. En raison de la nature même de l'interaction, il peut être difficile de conserver et de réutiliser les réponses.

Pour ce qui est de la dynamique, l'interaction peut être très amusante pour l'utilisateur. L'intensité de l'entrée et celle de la sortie peuvent être assez grandes, surtout si les entrées sont réduites au minimum, dans le cas des oui et non, par exemple. Il est évident que l'interaction présente un haut niveau de ludicité. L'utilisateur participe à un jeu en vue de gagner quelque chose. Pour chaque jeu, il n'y a habituellement qu'un seul protocole d'interaction, mais ce protocole peut être différent pour chaque jeu. Étant donné le découpage court et progressif de l'interaction, les réponses, tant du système que de la part de l'utilisateur, peuvent être données très rapidement. Les réponses du système devraient arriver à un rythme uniforme puisque, à

chaque étape, il ne peut y avoir qu'un nombre relativement faible de possibilités, particulièrement si les questions n'exigent qu'un oui ou un non pour réponse. L'utilisateur peut avoir ou non l'impression de maîtriser le système selon qu'il a l'impression de gagner ou de perdre. Si le jeu a été conçu de façcn à gagner assez facilement, l'utilisateur aura alors l'impression d'exercer un contrôle sur le système.

4 BESOINS EN MATIÈRE D'INTERROGATION

Dans une section précédente, nous avons divisé le processus d'interrogation en trois parties: la demande, la réponse et la dynamique de l'interaction. En ce qui concerne l'interaction utilisateur, c'est la demande et sa dynamique qui constituent la partie la plus difficile. En effet, lorsque l'utilisateur établit une demande, il essaie d'informer le système de ses besoins précis. Il doit fournir un grand nombre de renseignements et faire preuve d'une grande exactitude dans sa demande, s'il veut que le système l'interprète correctement. D'autre part, l'utilisateur a beaucoup plus de facilité à comprendre les réponses du système, même si la présentation de celles-ci laisse à désirer. Ceci s'explique du fait que les humains sont beaucoup plus intuitifs et qu'ils s'adaptent bien plus facilement que les systèmes informatiques. Cela ne signifie pas que les aspects de réponse à l'interrogation ne sont pas importants. Cependant, d'ici la fin du rapport, nous allons surtout nous pencher sur les aspects de demande du langage d'interrogation et de sa dynamique.

Tant pour les aspects de demande que pour les aspects de dynamique d'un langage d'interrogation, certaines caractéristiques ont été établies pour décrire les paramètres de l'interaction utilisateur. À partir de ces caractéristiques et de leur évaluation par rapport aux différents langages d'interrogation, nous pouvons fixer certaines caractéristiques dont devrait être doté, idéalement, un langage d'interrogation de base de données externes.

En général, la personne qui veut avoir accès à des bases de données externes ne sait pas dactylographier et ne veut pas être obligé de le faire pour être en mesure d'utiliser un langage d'interrogation. Donc, il faudrait qu'un nombre minimal de frappes permettent d'entrer en communication avec le système. Il faut absolument disposer d'un pointeur pour désigner les parties voulues de l'affichage. Ce pointeur

devrait être mécanique - un manche à balai ou une souris, par exemple - pour que l'utilisateur puisse se concentrer librement sur l'écran pendant qu'il s'en sert. Selon toute vraisemblance, il sera impossible d'éliminer le travail de dactylographie. Donc, il faut aussi que le système tolère et corrige les erreurs de dactylographie et que l'utilisateur puisse avoir recours à des pseudonymes pour faire des renvois à des objets dans des bases de données externes.

Toutes les commandes nécessaires pour indiquer au système qu'il doit intervenir doivent pouvoir être données au moyen de touches de fonction et de menus et être nommées d'après la fonction qu'elles permettent d'exécuter. En outre, il importe de restreindre à un minimum le nombre de ces commandes. Ainsi, l'utilisateur n'aura pas à se souvenir des commandes disponibles et pourra facilement se rappeler leurs fonctions d'après les noms.

Le système doit aider l'utilisateur à formuler sa demande. À cette fin, le système doit présenter les options offertes et, aussi, fournir des instructions en direct à tout moment au cours de l'interaction. La façon dont les demandes sont formulées doit faire appel à l'intuition de l'utilisateur. À cet égard, il devrait pouvoir disposer d'un paradigme facile à comprendre et qui l'aiderait à se souvenir des protocoles d'interaction. De plus, le système peut fournir des indices à l'utilisateur - au moyen de l'affichage, par exemple - qui l'aident à formuler correctement ses demandes.

En cherchant à aider l'utilisateur à formuler ses demandes, nous devons éviter de déterminer entièrement à l'avance les demandes qui peuvent être formulées, ce qui enlèverait toute liberté d'action à l'utilisateur. Le langage d'interrogation doit laisser une certaine latitude pour ce qui est de la sélectivité des demandes. À cette fin, le langage d'interrogation doit au moins permettre à l'utilisateur

de préciser le contenu de sa demande, si ce n'est sa forme exacte.

Dans la plupart des langages d'interrogation, l'utilisateur est censé connaître la structure ou le schéma de la base de données. Pour les bases de données externes, nous ne pouvons pas présumer que l'utilisateur connaîtra ou même comprendra la structure de la base de données ou la différence entre le schéma et la base de données. Il devrait plutôt y avoir une méthode permettant à l'utilisateur de passer facilement de la structure aux données. En outre, la façon dont l'utilisateur interroge la structure d'une base de données doit s'apparenter le plus possible à celle qu'il utilise pour interroger la base de données même. De cette façon, l'utilisateur n'a besoin d'apprendre qu'un seul jeu de protocoles d'interaction qui s'applique à toutes ses interactions avec le système.

Des études montrent que l'utilisateur du système informatique peut être fort désorienté s'il ne sait pas où il se trouve dans le système (Mantei, 1981). En outre, on ne peut s'attendre à ce qu'un utilisateur se souvienne de tous les éléments qui interviennent dans une interaction. Afin de donner à l'utilisateur des points de repère, nous devrions le plus possible conserver sur l'écran l'itinéraire suivi dans l'interrogation de la base de données (ou au moins en disposer pour en faire l'affichage au besoin).

Les interfaces d'utilisateur qui servent à consulter des données non mises en forme sont souvent fort différentes de celles qui servent à consulter des données mises en forme. Lorsque des bases de données externes sont interrogées, plusieurs types de données différents peuvent se présenter. La plupart des données se présenteront probablement sous une forme qui fera que l'utilisateur ne pourra pas distinguer les

données mises en forme (renseignements sur le stock, par exemple) des données non mises en forme (information sur les ventes par catalogue, par exemple). En outre, il n'est probablement pas souhaitable d'exiger un jeu distinct de protocole pour des types de données différents. Ainsi, l'interface de l'utilisateur doit être le plus uniforme possible pour l'interrogation des différents types de données que l'on trouve dans les bases de données externes.

Même si nous fournissons des protocoles d'interaction uniformes pour tous les types de demandes, nous pouvons admettre une interface d'utilisateur peu linéaire pour ce qui est de la difficulté éprouvée dans la formulation de la demande. Autrement dit, il est très facile de formuler des demandes simples, tandis qu'il peut être extrêmement difficile de formuler des demandes plus compliquées. En fait, on peut supposer que des sommes logiques et autres demandes complexes ne se présentent pas fréquemment. Quoi qu'il en soit, c'est la performance qui laissera à désirer lorsque nous exécuterons ces opérations. Pour que l'utilisateur puisse faire ces interrogations, il lui faudra une interface évoluée et spécialisée qui donne accès aux mêmes données que l'interface pour non-spécialistes.

Finalement, l'interface d'utilisateur devrait tirer profit de l'aptitude des gens à se souvenir d'abstractions. En outre, on devrait avoir recours à la technologie disponible (le graphisme et la couleur) pour faire la distinction entre les différents types d'objets visualisés à l'écran. Par exemple, nous devrions faire la distinction entre les données que l'utilisateur recherche, les données que l'on peut choisir et celles qui sont fournies par le système. Nous n'insisterons pas davantage sur cet aspect, mais il peut revêtir beaucoup d'importance puisqu'il guide l'utilisateur dans ses interactions avec le système. Il faut donc lui accorder une attention spéciale au moment de décider de la disposition de l'affichage qui sera présentée à l'utilisateur.

À partir de ces considérations comme principe directeur de la formulation des demandes et de l'interaction utilisateur, nous allons esquisser un langage d'interrogation des bases de données externes dans les pages qui suivent. Nous présentons d'abord, pour le langage d'interrogation, un paradigme qui véhicule par un exemple facile à comprendre l'essence de l'interaction de l'utilisateur, ensuite une vue d'ensemble de la conception d'un langage d'interrogation et finalement, une analyse des facteurs de conception de ce langage.

5 PARADIGME

Pour avoir accès à des informations au moyen d'un langage d'interrogation interactive de bases de données externes, l'utilisateur doit disposer d'un paradigme qu'il peut suivre et comprendre facilement. Ce paradigme doit aider l'utilisateur à se souvenir des protocoles d'interrogation (et à les connaître). Le paradigme que nous avons choisi est celui de la navigation d'un vaisseau spatial personnel dans l'espace.

Notre vaisseau spatial est très puissant et sa conception est des plus récentes. Il nous permet de voyager avec une facilité relative tant sur de longues distances - entre les galaxies - que sur de courtes distances - entre les planètes voisines. Le but de notre voyage dans l'espace est de parvenir à une planète et de l'explorer, quoique nous puissions aussi nous borner à une exploration de l'espace en soi. Une fois parvenus à une planète, nous pouvons l'explorer en la survolant ou en nous y posant et en examinant plus en détail une région donnée.

L'espace est très vaste et complexe et ne nous est généralement pas familier. Notre vaisseau est équipé d'un écran qui nous permet de voir l'espace dans toutes les directions, mais seulement une partie à la fois. Il est très difficile de naviguer sans aides, c'est-à-dire sans instruments, quoiqu'il puisse être préférable de procéder ainsi lorsqu'on explore une partie de la planète. Pour faciliter nos déplacements, notre vaisseau est équipé d'instruments qui peuvent dessiner des cartes dans le détail. Ces cartes peuvent prendre différents niveaux d'abstraction: structure générale d'une galaxie ou structure générale d'une planète, avec niveaux intermédiaires d'abstraction. On peut les projeter sur l'écran de visualisation du vaisseau.

Nous pouvons naviguer de plusieurs façons. Si nous avons une destination en vue, nous pouvons la préciser et notre vaisseau nous y amènera automatiquement. D'autre part, si nous ne faisons qu'explorer l'espace, il se peut que nous ne sachions pas exactement quelle sera notre prochaine destination. Nous pouvons nous en faire une idée ou accorder de l'intérêt à toute exploration interplanétaire. Si nous voulons savoir quelles destinations sont à notre portée, nous n'avons qu'à lire les cartes présentées sur l'écran de visualisation. Nous pouvons demander l'affichage de certaines cartes présentant une ou quelques propriétés particulières, ou toutes les cartes. En général, au fur et à mesure que nous envisageons les différentes destinations possibles, nous consulterons plusieurs cartes, des plus générales aux plus précises (en consultant, dans cet ordre, des cartes de la galaxie, des étoiles, puis des planètes). Ensuite, nous choisirons une destination et nous programmerons les instruments pour que notre vaisseau nous y conduise.

De la même façon nous pouvons imaginer qu'il existe des sources d'information auxquelles nous désirons avoir accès et qui se trouvent dans un **espace informationnel**. La composition de cet espace informationnel est modelée sur celle de l'espace même. Cet espace se compose d'objets abstraits comme des galaxies et des systèmes stellaires et d'objets concrets comme les planètes. Les objets abstraits de l'espace informationnel correspondent à la **structure** (schéma) des données. La structure nous renseigne d'une certaine façon sur la nature des données auxquelles nous pouvons avoir accès. La partie concrète correspond aux **données** (bases de données) de l'espace informationnel. La partie "données" correspond au niveau le plus bas des détails que nous pouvons explorer dans l'espace informationnel.

Grâce à notre vaisseau (système informatique), nous pouvons explorer l'espace informationnel. Le poste de commandement de notre vaisseau est constitué par le langage d'interrogation et notre écran de

visualisation par l'écran cathodique (de télévision) qui nous permet d'observer l'espace informationnel. Tout comme nous ne pouvons voir qu'une partie de l'espace en un temps donné sur l'écran de visualisation de notre vaisseau, notre écran cathodique ne nous présente qu'une partie de l'espace informationnel à un moment précis. L'écran cathodique définit les limites de l'espace informationnel qui sont observables comme l'écran de visualisation de notre vaisseau définit la partie de l'espace visible. Nous dirons de l'écran cathodique que c'est notre **portique** sur l'espace informationnel puisqu'il sert de fenêtre d'observation et nous permet de voir uniquement les informations auxquelles la fenêtre donne accès.

Les manoeuvres dans l'espace informationnel reposent sur trois actions essentielles, comme pour les manoeuvres faites pour se déplacer dans l'espace extra-atmosphérique. Tout d'abord, nous devons pouvoir afficher dans notre portique les structures (cartes) de l'espace informationnel qui nous intéressent, afin de choisir notre destination. Deuxièmement, nous devons pouvoir nous rendre à destination, c'est-à-dire avoir accès à la source d'information qui présente un intérêt. Finalement, nous devons être en mesure d'explorer les données qui se trouvent dans la source d'information choisie.

Deux remarques s'imposent au sujet de ce paradigme. D'une part, nous séparons distinctement la structure des données sur le plan conceptuel mais, sur le plan opérationnel, (comme nous allons le voir), structure et données sont traitées de la même façon (ou presque). Cette séparation pourrait se révéler importante à l'avenir puisque la structure pourrait résider dans le terminal de l'utilisateur tout comme les cartes précédentes sont gardées dans le vaisseau spatial. Les données se retrouvent dans une source d'information et, pour y avoir accès et les explorer, l'utilisateur doit s'y rendre tout comme il doit se rendre sur une planète pour l'explorer en détail. D'autre part, par la façon dont nous avons décrit l'interaction de l'utilisateur avec le

système, la plupart des modalités d'accès à l'information se règlent automatiquement. L'utilisateur fournit certains points de repère, mais c'est le système qui décide de la façon dont les choses se déroulent. L'accès automatique est préférable pour le profane, mais le spécialiste pourrait le trouver ennuyeux. L'utilisateur pourrait prendre en main les commandes et diriger le système plus directement, tout comme nous pourrions diriger notre vaisseau manuellement. L'utilisateur expérimenté disposerait ainsi d'un langage d'interrogation plus puissant et plus complexe. Dans ce rapport, nous examinons uniquement le langage d'interrogation dit automatique, mais il ne faut pas qualifier d'impossible l'élaboration d'un langage d'interrogation "manuel" qui compléterait le précédent.

6 VUE D'ENSEMBLE DE LA CONCEPTION

La présente section, décrit la conception globale d'un langage d'interrogation interactif servant à donner accès à des bases de données externes. Nous allons voir **ce qui** peut être fait et non **la façon** de le faire. Ainsi, nous allons décrire les étapes de façon générique et utiliser des terme génériques comme "pointer", "remplir" et "choisir" sans tenir compte de la façon dont ces actions se font. Nous aurons recours à l'analogie établie dans la section précédente pour expliquer les opérations en langage d'interrogation. Nous allons examiner ensuite quelques-uns des moyens dont dispose l'utilisateur pour exécuter ces opérations.

6.1 Choix d'une base de données externe

Supposons que nous sommes dans notre vaisseau spatial quelque part dans l'espace et que nous voulons choisir une destination. À l'aide de notre écran de visualisation, nous pouvons faire un balayage de l'espace et observer les étoiles. Cependant, tout ce que cela nous permet de voir, c'est une représentation abstraite. Le pilote chevronné peut choisir une destination directement à partir de la position des étoiles et mettre le cap sur cette destination. Cependant, en général, nous ne serons probablement pas familiarisés avec l'espace. Pour choisir une destination, nous pouvons faire le balayage de différentes parties de l'espace et consulter des cartes représentant chacune de ces parties. Nous pouvons demander à voir certaines parties de l'espace qui présentent des propriétés particulières ou nous pouvons observer toutes les parties de l'espace. Pendant la consultation des cartes, nous pouvons demander une autre carte plus détaillée d'une région donnée. Nous pouvons aussi faire des annotations dans le cas de certaines régions qui présentent un intérêt particulier et sur lesquelles on voudrait revenir. Plus tard, nous déciderons de la destination avant de fixer le cap.

Imaginons maintenant que nous essayons de consulter une base de données externe. La première fois que nous entrons en communication avec le système, nous ressemblons au passager du vaisseau spatial. Nous sommes dans l'espace extra-atmosphérique et, du portique, nous pouvons voir une représentation abstraite (modèle) des bases de données qui nous sont offertes. L'image est comme le balayage de l'espace qui nous montre simplement les configurations stellaires sans autre information sur ce qu'elles représentent. Par exemple, lorsque nous sommes dans le portique, nous pouvons observer un modèle semblable à celui de la figure 1.

Le format exact du modèle dans le portique ne revêt pas d'importance ici (une étude permettrait de déterminer la meilleure présentation). Il importe de retenir que le modèle comprend trois parties essentielles. Une de ces parties précise le **trajet** suivi pour arriver à ce modèle dans le portique. Dans notre exemple, ce renseignement se retrouve dans la zone ITINÉRAIRE. Une deuxième partie sert à la **spécification** et à **l'affichage des données.** Dans notre exemple, ces renseignements correspondent aux zones CATÉGORIE et CONTENU. Finalement, il y a une partie réservée à la **rétroaction du système** que nous avons nommée MESSAGE.

SERVICE D'INFORMATION VIDÉOTEX

ITINÉRAIRE:_____

CATÉGORIE	**CONTENU**
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

MESSAGE:_____

Figure 1: Exemple de modèle

Si nous ignorons à quelle base de données externe nous voulons avoir accès, nous pouvons demander au système d'afficher tous les renseignements dont il dispose sur les bases de données à ce niveau. Nous obtenons alors l'affichage de toutes les entrées qui correspondent au modèle. Par exemple, nous pourrions obtenir un affichage comme celui illustré à la figure 2 si nous avons demandé l'affichage de toutes les entrées.

SERVICE D'INFORMATION VIDÉOTEX

ITINÉRAIRE: _____

CATÉGORIE	CONTENU
Divertissements	Films
	Spectacles
	Expositions
	Sports
	Télévision
Nouvelles	Nouvelles
	- internationales
	- nationales
	- régionales
	Nouvelles du sport
	Divertissements
	Météo
Restaurants	Chinois

MESSAGE: Première de dix images

Figure 2: Exemple d'affichage

Nous pouvons maintenant examiner la carte d'information, et marquer certaines entrées pour référence ultérieure en les pointant ou en en choisissant une par pointage. Le choix d'une entrée nous permet d'avoir accès à des renseignements plus précis. Par exemple, si l'on choisit la catégorie "divertissements" et si l'on demande l'affichage de toutes les entrées comprises dans cette catégorie, nous obtiendrons un affichage comme celui de la figure 3.

Il se peut aussi que nous soyons intéressés par certaines bases de données, mais que l'on en ignore l'emplacement. Dans ce cas, nous pouvons demander au système de faire apparaître plus d'informations sur ces destinations en pointant et en remplissant la zone CATÉGORIE. Par exemple, nous pouvons introduire l'élément "divertissements" dans la zone CATÉGORIE et en demander l'affichage du contenu comme à la figure 4.

SERVICE D'INFORMATION VIDÉOTEX

INTINÉRAIRE: Divertissements

CATÉGORIE	**CONTENU**
Films	Premières
	Revues
	Festivals
Spectacles	Théâtre
	Ballet
	Opéra
	Récitals
Expositions	Art
	Livres
	Artisanat
Sports	Hockey
	Soccer

MESSAGE: Première de deux images

Figure 3: Exemple d'affichage

Par la suite, nous pouvons examiner la carte d'information, marquer certaines entrées pour référence ultérieure et(ou) en choisir une en la

pointant. Par exemple, si nous choisissons l'entrée "divertissements" et demandons par la suite l'affichage des entrées, nous obtenons l'affichage illustré à la figure 3.

Plutôt que d'entrer de façon explicite une valeur de zone, nous pouvons indiquer une valeur en pointant la zone et en demandant l'affichage de toutes les valeurs. Nous pouvons alors considérer ces entrées comme éléments de remplissage de cette zone. Dans ce cas, plutôt que d'inscrire une valeur de façon explicite, nous le faisons de façon implicite en identifiant certaines valeurs parmi celles qui sont affichées. Ainsi, par exemple, nous pourrions préciser l'affichage illustré à la figure 4 en pointant et en demandant l'affichage des valeurs de la zone CATÉGORIE, en marquant l'entrée "divertissements" et en demandant un nouvel affichage.

SERVICE D'INFORMATION VIDÉOTEX

ITINÉRAIRE: _____

CATÉGORIE	**CONTENU**
Divertissements	Films
	Spectacles
	Expositions
	Sports
	Télévision
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

MESSAGE: Première et seule image

Figure 4: Exemple d'affichage

Finalement, si nous savons où nous voulons nous rendre, nous pouvons l'indiquer directement et le système se chargera de nous y amener. Pour ce faire, nous pointons une valeur pour la zone ITINÉRAIRE et nous l'y faisons entrer, ce qui indique au système qu'il doit mettre le cap sur la destination choisie et nous présenter un modèle de celle-ci.

Par des applications répétées de cette technique, nous pouvons naviguer dans la structure d'une base de données et arriver à l'information recherchée. Chaque fois que nous choisissons un élément

dans un modèle, nous obtenons des renseignements plus détaillés sur l'itinéraire conduisant à une base de données. Nous pouvons également conserver de l'information sur le trajet choisi pour en arriver au modèle en cours. Ceci revient à essayer de déterminer une destination pour notre vaisseau spatial. On peut avoir recours à cette technique à plusieurs niveaux, mais il est souhaitable que le nombre des niveaux soit restreint pour réduire la quantité d'interactions requises pour en arriver aux données. En général, le nombre de niveaux requis est déterminé par la complexité de la base de données à laquelle nous voulons avoir accès. Cet aspect correspond au but que nous nous sommes fixé d'assortir la complexité de l'accès à la complexité de l'information ou du type de demande formulée.

Au moyen des zones CATÉGORIE et CONTENU, nous nous ménageons en fait la possibilité d'utiliser des mots clés et des renvois. Autrement dit, toutes les bases de données ayant trait à une ou à des catégories déterminées seront affichées. De même, certaines données peuvent se retrouver dans plusieurs catégories. On note que les données affichées peuvent se retrouver à différents niveaux dans la structure. Autrement dit, certains peuvent renvoyer à des bases de données proprement dites, tandis que d'autres peuvent se rapporter à des parties d'une base de données. Cependant, l'utilisateur n'a pas besoin de connaître cette distinction. Ce qui apparaît dans le portique est fonction de la façon dont les renvois aux données ont été conçus dans le système.

Chaque mot-clé affiché dans le portique est accompagné d'une description plus détaillée de l'information représentée par ce mot. À tout moment, nous pouvons pointer un mot-clé et en demander la description, tout comme on peut consulter le guide qui accompagne une carte. Nous procédons ainsi uniquement lorsque nous voulons en connaître davantage sur un point donné.

La zone ITINÉRAIRE nous permet de voir le trajet suivi pour en arriver à l'emplacement actuel. On peut revenir sur ses pas en tout temps en pointant le niveau (trajet) où nous voulons nous retrouver.

6.2 Exploration des données

Supposons que notre vaisseau se dirige vers une planète. Une fois arrivés, nous pouvons survoler la planète et en faire l'observation ou nous y poser et en explorer une partie. Pour nous aider à naviguer, nous avons besoin de cartes. Ces cartes montrent la structure abstraite de la planète, sans entrer dans les détails. Nous pouvons explorer la planète en détail en la survolant ou en nous y posant.

De la même façon nous en venons à atteindre la partie "données" d'une base de données. Tout comme nous explorons et nous survolons la structure, nous pouvons maintenant explorer et survoler la partie "données". Comme dans le cas de la structure, une représentation abstraite (modèle) des données du portique nous est présentée ici. Un exemple de ce modèle apparaît à la figure 5.

SERVICE D'INFORMATION VIDÉOTEX

ITINÉRAIRE: Divertissements. Films. Premières

FILMS	**VEDETTE**	**CINÉMA**	**HEURES**	**TÉLÉPHONE**

MESSAGE:_____

Figure 5: Exemple de modèle

Nous choisissons des entrées de modèle particulières en pointant et en remplissant une zone. Par exemple, pour obtenir l'affichage de tous les films qui mettent en vedette Donald Sutherland, il suffirait de pointer la zone VEDETTE et d'y inscrire (la valeur) "Donald Sutherland". Le système nous indiquera alors tous les films qui mettent en vedette Donald Sutherland et qui sont à ce moment-là à l'affiche dans les cinémas. On peut également remplir une zone en la pointant et en demandant au système d'afficher toutes les valeurs qu'il possède pour

cette zone. Nous pouvons alors marquer certaines valeurs entrant dans cette zone. Le système nous donnera la liste complète de toutes les entrées où apparaît cette valeur. Par exemple, nous pourrions demander à voir toutes les valeurs correspondant à la zone VEDETTE. Nous n'aurions ensuite qu'à retenir pour examen ultérieur les valeurs qui nous intéressent.

Si nous ne voulons que parcourir un document, nous pouvons alors demander l'affichage de toutes les entrées du modèle. Par la suite, nous pouvons marquer certaines entrées particulières en les pointant, pour y revenir et en faire l'examen. Ceci permet d'éliminer les entrées qui ne nous intéressent pas et d'économiser l'espace d'affichage.

6.3 Itinéraires préétablis

Au cours de notre voyage en vaisseau spatial, nous découvrons sans tarder qu'il y a certaines destinations très fréquemment visitées. Nous en venons à connaître les itinéraires et nous pouvons rapidement nous passer de cartes. Une navigation à vue est possible.

De la même façon, lorsque nous avons accès à des bases de données externes, il y a certaines données que nous consultons très fréquemment. Pour accélérer les choses, nous pouvons établir à l'avance l'itinéraire que le système devra suivre. Nous pouvons nommer cet itinéraire et le préciser de façon explicite dans la zone ITINÉRAIRE, tout comme nous pointons directement notre destination sur les configurations stellaires visuelles que nous apercevons de notre vaisseau.

Ces itinéraires préétablis sont annexés à la structure du service d'information vidéotex, mais seul l'utilisateur qui les a définis y a accès. La destination peut être un point de la structure, les données ou une portion des données, c'est-à-dire un sous-ensemble de données.

De cette façon, on peut, dans une certaine mesure, personnaliser le résultat au profit de l'utilisateur en éliminant certains renseignements qui se trouvent dans la partie donnée, par exemple. On peut distinguer les noms des itinéraires préétablis du nom des bases de données en utilisant des moyens appropriés pour les mettre en évidence.

7 CONSIDÉRATIONS EN MATIÈRE DE CONCEPTION

Dans la section précédente, nous avons vu ce que le langage d'interrogation peut faire, mais non pas de quelle façon il peut le faire. Nous avons eu recours à des termes génériques comme "pointer", "remplir" et "choisir" sans préciser de quelle façon ces actions doivent être faites. Dans la présente section, nous verrons plus en détail comment un utilisateur du service vidéotex peut réaliser ces actions.

7.1 Pointage

Une des actions les plus fondamentales dans les protocoles d'interaction que nous avons vus, c'est la possibilité de pointer un élément dans un portique en vue de le marquer ou de le choisir pour des actions ultérieures. Idéalement, ce pointage ne doit pas détourner l'attention de l'utilisateur de l'affichage dans le portique. Le pointage sonore serait de loin le plus souhaitable, mais il n'est peut-être pas encore réalisable sur les plans technique ou économique. Sur le plan qualitatif, le deuxième pointeur serait un pointeur mécanique comme une manche à balai, une rondelle ou une souris, l'utilisateur pouvant alors positionner un curseur à l'endroit voulu dans le portique et indiquer, d'une certaine façon, qu'il choisit cette position. (À noter qu'il est nécessaire de positionner les curseurs, mais aussi d'indiquer que c'est la position voulue.) Cette méthode de pointage est réalisable sur le plan technique et pourrait être rentable.

Faute d'avoir un pointeur pour préciser un emplacement arbitraire dans le portique, on peut toujours préciser des emplacements donnés, c'est-à-dire des zones particulières dans le portique. Cette précision devrait se faire aussi facilement que possible et tenir compte du fait que le positionnement est fonction de la structure du modèle affiché

dans le portique. Nous avons généralement besoin de pouvoir préciser une zone dans un modèle et peut-être une entrée dans une zone. Pour des raisons de généralité, ces deux précisions doivent être indépendantes l'une de l'autre. La sémantique exacte de la précision dépendra des caractéristiques du modèle affiché dans le portique.

La possibilité de choisir une zone et une entrée particulières conduit à la notion de zone et d'entrée "courantes". De façon visuelle, cette situation peut être représentée en esquissant à grands traits l'entrée et la zone courantes d'une certaine façon, c'est-à-dire par le clignotement, l'inversion de l'image ou l'utilisation d'une couleur donnée, par exemple. On peut préciser des opérations qui s'appliquent ensuite à l'entrée ou à la zone courantes comme nous verrons bientôt.

Pour la spécification des zones, il existe une opération de **zone** qui présente les options et la sémantique suivantes: **zone suivante** - permet de choisir la zone suivante; **zone précédente** - permet de choisir la zone précédente; **nom** de la **zone** - permet de choisir la zone dénommée.

Les zones d'un modèle comportent un ordre préétabli qui est utilisé par les options **zone suivante** et **zone précédente**. En outre, les zones se bouclent. La zone initiale est constituée par l'ensemble du modèle comme la zone qui vient après la dernière zone et avant la première. L'option **nom** fait appel à un nom symbolique pour faire directement référence à une zone. Il faut établir des conventions pour nommer les zones de façon symbolique. Certaines fonctions (définies ci-dessous) ont besoin d'une zone comme paramètre. Si aucune zone n'est précisée de façon explicite, c'est l'ensemble du modèle qui joue ce rôle. Ainsi, lorsqu'une zone n'est pas expressément définie, le mode **affichage** renvoie à l'affichage de toutes les valeurs d'un modèle,

tandis que le mode **affichage** qui suit le choix d'une zone renvoie au seul affichage des valeurs correspondant à la zone choisie.

Pour la spécification des entrées, il existe une opération **entrée** qui permet de choisir une entrée d'une zone à l'aide des options et de la sémantique suivantes: **entrée suivante** - permet de choisir l'entrée suivante; **entrée précédente** - permet de choisir l'entrée précédente: **nom d'entrée** - permet de choisir l'entrée appelée.

Les entrées présentent un ordre préétabli qu'utilisent les options entrée **suivante** et entrée **précédente**. On peut aussi faire référence à des entrées au moyen de noms symboliques qui permettent de choisir directement une entrée. Par exemple, pour une liste des valeurs d'une zone, on peut désigner symboliquement les entrées par un, deux, etc. Si la zone courante est constituée par le modèle, une entrée renvoie alors à toutes les valeurs qui définissent un des cas du modèle.

7.2 Positionnement dans le portique

Jusqu'à maintenant, nous n'avons tenu compte d'aucune restriction quant aux dimensions du modèle qui pourrait être affiché dans le portique. Nous avons supposé que tout élément pouvait être affiché immédiatement. Cependant, il se peut que les modèles soient trop encombrants pour entrer dans le portique. Ainsi, les dimensions de notre portique restreignent le nombre de données que nous pouvons afficher en un temps donné. Idéalement, nous aimerions pouvoir déplacer le portique continuellement pour suivre le modèle comme s'il s'agissait d'une caméra de télévision.

Dans les systèmes vidéotex, l'espace informationnel est habituellement divisé en unités de dimensions fixes (pages) correspondant à celles du portique (écran de télévision) et un mouvement

continu dans toutes les directions est impossible. Par conséquent, nous pouvons supposer que les modèles sont plutôt divisés en **images** de **dimensions** fixes. Une image est une partie de l'espace informationnel que nous pouvons observer dans le portique à un moment donné (tout comme on peut regarder une diapositive à l'aide d'un projecteur). En général, on peut, à partir d'une image, passer à n'importe quelle autre image (contrairement à ce qui se produit dans le cas d'une projection de diapositives où seules la diapositive précédente ou la diapositive suivante peuvent être vues).

Une image renferme uniquement le nombre de données nécessaires pour remplir le portique. Les images qui se trouvent au-dessus et au-dessous de l'image observée contiennent, respectivement, les entrées de modèle précédentes et suivantes. Celles de droite et celle de gauche contiennent des entrées supplémentaires qui ont trait à l'image observée, mais qui ne peuvent être affichées immédiatement dans le portique. Ainsi, par exemple, des tables de données trop larges pour entrer dans le portique peuvent être réparties en plusieurs images adjacentes verticales (à la droite et(ou) à la gauche l'une de l'autre). Des tables de données trop longues pour entrer dans le portique peuvent être divisées en plusieurs images adjacentes horizontales (l'une au-dessus (ou) au-dessous de l'autre).

Pour amener le portique en position avec l'image voulue, il existe une opération **image** avec les options et la sémantique suivantes: **image supérieure** - permet de déplacer le portique jusqu'à l'image qui se trouve au-dessus de l'image observée; **image inférieure** - permet de déplacer le portique jusqu'à l'image qui se trouve au-dessous de l'image observée; **image de gauche** - permet de déplacer le portique jusqu'à l'image qui se trouve à la gauche de l'image observée; **image de droite** - permet de déplacer le portique jusqu'à l'image qui se trouve à la droite de l'image observée; **nom de l'image** - permet de déplacer le portique jusqu'à l'image appelée.

On peut, symboliquement, donner un nom aux images pour qu'il soit possible d'amener directement le portique sur ces images. Dans la plupart des cas, et probablement au début, il est souhaitable de ne pas toucher aux images de droite et de gauche puisque ceci aurait pour effet de fragmenter considérablement les données. Dans ce cas, la fonction **image** se réduit aux fonctions **image supérieure, image inférieure** et **nom de l'image.**

S'il est nécessaire de faire intervenir les images de gauche et de droite, il est alors souhaitable d'avoir recours à une certaine méthode pour faire connaître à l'utilisateur sa position (l'orienter). Cette orientation comporte deux aspects. Tout d'abord nous cherchons à informer l'utilisateur sur sa position par rapport à l'ensemble de l'espace informationnel visible à ce niveau. Une technique utilisée à cette fin consiste à illustrer graphiquement la position du portique par rapport à l'ensemble de l'espace informationnel. Par exemple, nous pourrions montrer l'espace informationnel dans son ensemble sous la forme d'un polygone et la position du portique sous la forme d'un polygone superposé au polygone plus grand qui représente l'espace informationnel. Comme autre solution, nous pourrions utiliser la zone du message pour indiquer le déplacement de l'utilisateur vers la gauche ou vers la droite par rapport à un point de référence donné, comme nous avons illustré le déplacement à la verticale dans les exemples précédents.

Le deuxième aspect qui doit nous préoccuper, c'est le maintien d'un lien sémantique entre les différentes images qui se trouvent dans le portique. Dans le plan vertical, la question ne se pose pas vraiment, étant donné qu'en faisant un balayage ascendant et descendant, la même partie du modèle est affichée dans le portique. Cependant, dans le plan horizontal, la partie du modèle visible dans le portique change au fur et à mesure de nos déplacements vers la gauche et vers la droite. Il est donc souhaitable de conserver un certain point commun entre les

images adjacentes horizontales. Pour ce faire, on peut choisir une des zones de modèle comme **zone clé** et la garder dans le portique pendant toute la durée de notre déplacement horizontal.

Par exemple, à la figure 5, une des zones du modèle, disons la zone FILMS, pourrait demeurer dans le portique pendant que nous déplaçons la position du portique dans le plan horizontal.

7.3 Fonctions de commande

Les fonctions de commande servent à signaler au système qu'il doit fournir des renseignements sur son état et prendre des mesures appropriées. Le recours à une fonction de commande peut être vu comme le recours à une procédure qui prend comme entrée l'état ponctuel du système et(ou) la valeur de la zone. La valeur des paramètres d'entrée donne lieu à certaines actions. Les fonctions de commande peuvent prendre différentes formes, par exemple, boutons de commande ou commandes de menu.

CHOIX

La fonction **choix** prend comme paramètre d'entrée la valeur de la zone courante. Le résultat de la fonction dépend des caractéristiques de la valeur de la zone. Si nous choisissons une valeur de la zone "catégorie", celle-ci devient partie intégrante de l'itinéraire que nous avons emprunté. Si nous choisissons un nom dans la zone ITINÉRAIRE, le système nous ramène à ce point. Comme nous allons le voir, si nous choisissons une valeur de données qui est du type **image** ou **son**, nous obtenons l'affichage d'une image, la présentation d'un film ou d'une bande magnétoscopique ou la lecture de messages sonores enregistrés. S'il n'existe pas de procédure explicite définie pour une valeur de zone, le fait de la choisir ne produit aucun effet et il en résulte donc une action nulle.

REPÉRAGE

La fonction **repérage** prend comme paramètre d'entrée la valeur de la zone courante. Cette fonction donne au système l'ordre de prendre note de la valeur repérée pour référence ultérieure. Par exemple, lorsque nous examinons des valeurs de zone, nous pouvons repérer certaines zones pour en faire l'examen ultérieur. Le repérage ne peut se faire qu'en mode de spécification **d'affichage**.

DESCRIPTION

La fonction **description** prend comme paramètre d'entrée la valeur de la zone courante. Le système répond à cette fonction en fournissant une description détaillée de la valeur de la zone courante. Cette explication peut être visuelle, verbale ou prendre ces deux formes en même temps. Par exemple, pour une valeur de zone **catégorie**, la fonction **description** peut servir à obtenir une explication des renseignements qui se trouvent dans la catégorie. Les valeurs de zone n'exigent pas toutes une description. Donc, il n'y aura pas nécessairement de description lorsqu'on fait appel à cette fonction, c'est-à-dire qu'il y aura une action nulle.

AIDE

La fonction **aide** prend comme paramètre d'entrée l'état ponctuel du système. S'il fait appel à cette fonction, l'utilisateur reçoit des instructions sur les mesures qu'il peut prendre et sur la façon dont il peut les prendre compte tenu de l'état ponctuel du système.

ANNULATION

La fonction **annulation** prend comme paramètre d'entrée l'état ponctuel du système. Cette fonction donne au système l'ordre d'annuler la dernière commande ou le mode de spécification en cours.

INTERRUPTION

La fonction **interruption** prend comme paramètre d'entrée l'état ponctuel du système. Cette fonction conserve l'état ponctuel de l'interaction utilisateur. On peut relancer l'interaction utilisateur n'importe quand. L'interaction reprend alors à partir du point d'interruption, comme si elle n'avait jamais été interrompue. Les interactions interrompues sont conservées dans la base "interruption". On peut mettre fin à une interruption en choisissant ces interactions.

SORTIE

La fonction **sortie** prend comme paramètre d'entrée l'état ponctuel du système. Cette fonction met fin à l'interaction utilisateur en cours et aucune information sur cette interaction n'est conservée.

CONSERVATION

La fonction **conservation** prend comme paramètres d'entrée l'état ponctuel du système plus un nom fourni par l'utilisateur. Cette fonction conserve la spécification de l'interaction utilisateur et la dénomme de façon explicite. On peut ensuite copier l'interaction en choisissant le nom défini par l'utilisateur. Ce nom devient partie intégrante de l'information de la structure pour cet utilisateur. La fonction **conservation** permet de définir ce que l'on voit. Elle diffère de la fonction **interruption** puisqu'une interaction interrompue n'est conservée que jusqu'à ce qu'elle soit achevée.

D'autre part, une interaction conservée est gardée en permanence par le système. On peut laisser tomber une interaction conservée en gardant un état de système nul (initial) avec le même nom que l'état courant de système conservé.

7.4 Modes de spécification

Les modes de spécification ont trait à la façon dont un utilisateur spécifie des demandes de renseignements adressées au système. Lorsqu'on fait appel à un mode de spécification, une suite d'actions va suivre pour constituer la spécification. On peut mettre fin implicitement à un mode de spécification au moyen de la fonction **choix**, de la fonction **conservation** ou d'un autre mode de spécification, ou explicitement par la fonction **annulation** ou la fonction **sortie**. Pendant l'interaction avec des bases de données externes, l'utilisateur peut spécifier sa demande de deux façons. L'une de ces méthodes est le mode par défaut; c'est l'utilisateur qui précisera lequel. Le mode par défaut est introduit initialement lorsque l'utilisateur entre en communication avec le système et après chaque fonction **choix** ou **conservation**.

REMPLISSAGE

Le mode **remplissage** permet à l'utilisateur de choisir diverses zones et de préciser la ou les valeurs pour chacune. Il choisit les zones en les pointant. Ces dernières sont remplies explicitement au moyen d'une valeur qu'on y inscrit à partir d'un clavier ou, implicitement, en marquant des valeurs affichées comme on l'explique au mode **affichage**.

Si plusieurs valeurs sont précisées pour une zone donnée, on suppose alors que l'opération booléenne **ou** fait partie de ces valeurs (correspondance pour valeur 1 ou valeur 2, entre autres). Il est à noter qu'une opération booléenne **et** parmi ces valeurs est

impossible sauf si une entrée d'une zone peut effectivement prendre plusieurs valeurs. Si plus d'une zone est remplie, on suppose que l'opération booléenne **et** fait partie des zones choisies (correspondance pour zone 1 et zone 2, entre autres). On peut exécuter une opération booléenne **ou** comprise dans les zones si l'on remplit plusieurs modèles.

On indique la fin du remplissage d'un modèle en précisant le mode **affichage**, le modèle étant constitué par la zone en cours. L'utilisateur perçoit alors toutes les entrées du modèle qui correspondent à la spécification. Si aucune zone n'a été remplie, toutes les entrées correspondant au modèle sont affichées.

AFFICHAGE

Le mode **affichage** permet à l'utilisateur de voir toutes les entrées des zones ou des modèles dans le portique. Si le modèle est vide au moment où l'on fait intervenir le mode **affichage**, toutes les entrées sont affichées. Si le modèle est rempli, seules les entrées qui correspondent au modèle rempli sont affichées comme on l'indique au paragraphe où l'on décrit le mode de spécification **remplissage**.

En mode **affichage**, les entrées de modèle peuvent être explorées au moyen des opérations de pointage et de positionnement dans le portique. On peut choisir la zone et l'entrée courantes au moyen de la fonction **choix** qui fait appel à la procédure correspondant à la valeur de la zone en cours. On peut aussi marquer d'un repère les valeurs des zones en mode **affichage**. Consécutivement, si l'on passe de nouveau en mode **affichage**, seules les valeurs des zones marquées d'un repère seront choisies et affichées. Si la zone en cours est le modèle, seules les entrées repérées seront choisies en vue d'être affichées. Si la zone courante est une zone qui se trouve dans le modèle, seules les entrées qui correspondent aux valeurs des zones

repérées seront choisies. Les conventions du mode **remplissage** s'appliquent ici.

7.5 Types de données

La plupart des langages d'interrogation traditionnels ne traitent raisonnablement que des données mises en forme. Cependant, il existe d'autres types de données que l'on peut avoir à traiter lorsqu'on interroge des bases de données externes. En voici quelques-uns: données de textes mises en forme, données de textes non mises en forme, données d'images et données vocales.

En raison de la nature des différents types de données, le mode d'interrogation est propre à chaque type de données. Cependant, on peut utiliser les mêmes protocoles d'interaction pour tous les types.

DONNÉES DE TEXTES MISES EN FORME

Une fois atteinte la partie "données" (niveau d'exemple) dans le processus d'interrogation, le système nous présente un modèle du format des données. Un exemple de ce modèle est illustré à la figure 6.

Nous pouvons choisir des données dans un modèle semblable en utilisant les modes de spécification **remplissage** et (ou) **affichage** examinés précédemment. La valeur par défaut du remplissage de la zone suppose que nous voulons une correspondance parfaite pour la valeur précisée. Pour certains types de données, nous pouvons aussi prévoir d'autres sortes de correspondances: assortiments de structure (c'est-à-dire trouver une certaine structure dans une zone), inférieur à, inférieur ou égal à, supérieur à, supérieur ou égal à ainsi que concordances et discordances d'égalité. On peut indiquer

ces options en faisant précéder la valeur spécifiée d'un symbole approprié comme , , etc. Les zones qui doivent être remplies peuvent être choisies au moyen de l'opération de pointage examinée précédemment.

DONNÉES DE TEXTES NON MISES EN FORME

Si la partie données se compose de données de textes non mises en forme, nous obtenons alors la partie "données" qui nous est encore une fois présentée avec un modèle des données de textes non mises en forme (voir l'exemple à la figure 7).

Nous pouvons choisir des données dans un texte en précisant la structure recherchée, c'est-à-dire en indiquant dans la zone appropriée la structure du texte. À l'affichage de la structure spécifiée, le système nous montre la structure en contexte, c'est-à-dire l'endroit où elle apparaît dans le texte. Dans ce cas, les options **entrée suivante** et **entrée précédente** de l'opération **entrée** nous indiqueront les entrées suivantes et précédentes avec la structure en contexte (si la zone courante est constituée par le modèle). Nous pouvons aussi observer d'autres images avoisinantes au moyen de l'opération **image**. Tout le texte sera affiché si l'on passe en mode **affichage** sans remplir une formule.

SERVICE D'INFORMATION VIDÉOTEX

ITINÉRAIRE: Livres. Roman. Autant en emporte le vent. Chapitre 1

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

MESSAGE _____

Figure 7: Exemple de données non mises en forme

DONNÉES D'IMAGES ET DONNÉES VOCALES

Certaines zones de données de textes mises en forme ou non mises en forme peuvent correspondre à des données d'images ou à des données vocales. Actuellement, il est très difficile d'interroger ces types de données directement. À l'avenir, on pourra peut-être introduire comme données d'entrée une image ou un message vocal et demander au système tous les cas où on les retrouve. Cependant, pour l'instant, on doit se contenter d'afficher ou de reproduire une image ou les données vocales, en choisissant l'entrée appropriée d'une zone d'images ou d'une zone vocale au moyen de la fonction **choix.** Afin d'indiquer à l'utilisateur la nature de ces zones, on peut les faire ressortir au

moyen de couleurs particulières ou leur apposer des symboles graphiques spéciaux, par exemple l'icône d'un appareil photo pour indiquer des photographies.

8 REMARQUES ET CONCLUSIONS

Nous avons examiné le processus de l'interrogation interactive en général, puis relativement à des bases de données externes en particulier. Le processus d'interrogation a été divisé en trois parties: la demande, la réponse et la dynamique. Pour chaque partie, nous avons identifié plusieurs caractéristiques et indiqué des propriétés souhaitables. Nous avons ensuite identifié sept langages d'interrogation et évalué chacun d'après ces caractéristiques. À partir de cette évaluation, nous avons établi certaines propriétés dont devraient être dotés les langages d'interrogation de bases de données externes. Finalement, nous avons proposé un spécimen auquel devrait correspondre ces langages d'interrogation.

Pendant l'élaboration de ce spécimen, nous avons fait porter nos efforts sur les aspects de requête et de dynamique du langage d'interrogation. Dans le cas des demandes, les paramètres de l'interaction utilisateur étaient les suivants: nombre de frappes requises, type et nombre de commandes disponibles, degré de liberté dans la formulation des demandes, sélectivité des demandes, uniformité des demandes par rapport aux bases de données consultées et aptitude à personnaliser la formulation de la demande relativement à une demande particulière. Dans les cas de la dynamique de l'interaction, les paramètres étaient les suivants: intensité entre l'utilisateur et le système, degré de ludicité de l'interaction, nombre de protocoles disponibles pour l'interaction, réaction du système à une demande de l'utilisateur et degré de contrôle de l'utilisateur dans une interaction. Voyons maintenant de quelle façon ces propriétés s'intègrent au spécimen proposé.

Le nombre de frappes requises pour préciser une demande dépend du mode de spécification choisi par l'utilisateur. Si celui-ci affiche et

marque des entrées, il lui suffit de pointer, marquer et choisir des valeurs. Le nombre de frappes nécessaires dépendra du pointeur disponible. Le marquage nécessite une frappe pour chaque valeur, tout comme la sélection. Si on a recours au mode remplissage, l'utilisateur peut alors pointer les zones et y faire entrer des valeurs. Cependant, même dans ce cas, de nombreuses frappes peuvent être évitées si l'utilisateur affiche les valeurs de la zone et les marque.

Pour le spécimen proposé, il y a deux modes de spécification, huit *opérations de service* et peut-être (selon le pointeur) de un à trois pointages. Un des deux modes de spécification sera le mode par défaut, que l'utilisateur n'aura pas besoin de préciser explicitement. Des huit opérations de service, il y a en deux dont on fera probablement un usage intensif: l'opération **choix** et l'opération **marquage**. Si des boutons portant le nom de la fonction correspondante sont prévus, l'utilisateur n'a pas à faire d'effort pour se souvenir du nom des fonctions. En outre, le nom des fonctions se veut représentatif et on peut toujours obtenir une aide en direct au moyen de la fonction **aide**.

En raison de la façon dont l'interaction utilisateur a été conçue, il est très difficile de formuler une demande incorrectement, que ce soit sur le plan de la syntaxe ou sur celui de la sémantique. Sur le plan de la syntaxe, le langage d'interrogation ne comprend pas de syntaxe dont il faut se souvenir. Les modes de spécification acceptent, pour les modèles, tout état d'entrée: vide, à moitié plein et plein à capacité. Les opérations de services sont toujours valides, mais elles peuvent donner lieu à un résultat nul. Sur le plan de la sémantique, les noms des zones apparaissent toujours dans le portique et l'utilisateur peut voir la structure des données au moyen du modèle. On peut donner des précisions inexactes sur une demande compliquée, c'est-à-dire s'il y a de nombreuses zones à qualifier au moyen de **et** et **ou** par exemple. Cependant, la façon dont les zones d'un modèle

interagissent par défaut (**et** entre les zones **ou** à l'intérieur des zones) est assez intuitive. On doit accepter que des demandes compliquées sont complexes et que leur spécification est plus facilement erronée.

La sélectivité du langage d'interrogation est presque aussi bonne que les mots-clés et les langages d'interrogation par exemple. Les sommes logiques sont les seules opérations difficiles à préciser (en fait nous n'avons pas dit de quelle façon les exécuter). Un des moyens consiste à utiliser une interaction conservée comme entrée d'une autre interaction. Cependant, nous ne croyons pas que la demande d'un langage d'interrogation de base de données externes permettant de résoudre des sommes logiques sera généralisée. Quoi qu'il en soit, on peut toujours avoir recours à une interface d'utilisateur plus expérimentée (comme un langage d'interrogation par mots-clés) pour ces demandes.

Dans le langage d'interrogation proposé, nous n'avons pas fait de supposition quant à la structure des bases de données externes. La nature de l'interface choisie - modèles - ne présuppose pas de structure de base de données sous-jacente particulière. Ainsi, on peut utiliser le langage d'interrogation pour assurer l'interface avec toute base de données, pourvu que des algorithmes appropriés soient prévus pour traduire les opérations et les structures du langage d'interrogation en opérations et structures du SGBD (système de gestion de base de données) de la base de données externe. On fait actuellement de nombreuses recherches dans ce secteur et plusieurs approches de la question peuvent être prises en considération (Date, 1980; Vassiliou and Lochovsky, 1980).

Le langage d'interrogation peut facilement s'adapter à des applications particulières. Il suffit, pour ce faire, de personnaliser les modèles dont se sert l'utilisateur pour interagir avec le système. Aucun autre aspect du langage d'interrogation n'a besoin d'être modifié. Plus particulièrement, le matériel requis pour l'interaction et les

opérations de pointage, les opérations de services et les modes de spécification demeurent les mêmes.

L'intensité de l'interaction dépend du mode de spécification choisi. Pour l'affichage et le marquage, on peut avoir une interaction très intense. Pour le remplissage des modèles, elle peut être modérée. Il est à noter que les concepts sur lesquels repose l'interaction demeurent les mêmes lorsque nous passons du pointage au moyen d'un clavier à celui utilisant un dispositif mécanique, puis à un pointage au moyen d'instructions vocales. Ainsi, le langage d'interrogation peut tirer profit des progrès technologiques qui permettent d'accroître l'intensité de l'interaction.

La ludicité du langage d'interrogation varie de modérée à bonne. L'utilisateur peut explorer facilement d'autres trajets de la structure sans grand effort étant donné qu'il peut facilement conserver les trajets et(ou) rebrousser chemin en tout temps. Ce processus d'exploration permet à l'utilisateur de s'approcher graduellement de son but et de voir le chemin parcouru au moyen de la zone ITINÉRAIRE du portique. Selon la façon dont sont construits les renvois, l'utilisateur peut accéder aux mêmes données à partir de trajets différents. Le pointage, le marquage, l'affichage et le remplissage sont des types d'opération où l'utilisateur participe à l'interaction. L'utilisateur n'a pas seulement à appuyer sur un bouton. La conception des modèles et l'utilisation de couleurs et de sons peut ajouter à la ludicité du langage d'interrogation.

Il existe plusieurs protocoles permettant d'interroger les bases de données: le remplissage, l'affichage et les itinéraires préétablis. L'utilisateur peut choisir celui qui lui semble le mieux adapté à l'interaction donnée et à son niveau de compétence. Il peut aussi opter pour le mode de spécification par défaut selon la connaissance qu'il a du système.

Compte tenu de la technologie actuelle, la réponse du système peut être un facteur limitant le langage d'interrogation. Plus particulièrement, en mode **affichage**, le système doit afficher rapidement l'information demandée. Sur le plan des données, cela peut signifier l'extraction d'un grand nombre de renseignements. Pour qu'il y ait affichage des valeurs d'une zone quelconque, on doit disposer d'un mécanisme d'accès rapide pour extraire promptement toutes les valeurs. Au début, il faudra peut-être restreindre l'utilisation du mode **affichage** à la structure et à certaines zones des données pour surmonter les problèmes de réponse. Cependant, il se pourrait bien que les progrès technologiques viennent bientôt abolir cette restriction. Finalement, toute la nature de l'interaction de l'utilisateur - commander des affichages au système, marquer, etc. - contribue à donner à l'utilisateur le sentiment qu'il maîtrise la situation. C'est lui qui dirige le système pour son compte et pour atteindre son but. De plus, le paradigme d'interrogation que nous avons examiné précédemment contribue à créer ce sentiment. Il donne à l'utilisateur l'impression que le système est son allié puisqu'il l'aide à guider son vaisseau spatial (le système) dans un milieu hostile (l'espace informationnel) et qu'il le protège contre les dangers qui le guettent.

BIBLIOGRAPHIE

Bonczek, R.H., Cash, J., et Whinston, A. (1977). "A Transformational Grammar-based Query Processor for Access Control in a Planning System," ACM TODS 2, pp. 326-338.

Bonczek, E.H., et Whinston, A. (1977). "A Generalized Mapping Language for Network Data Structures," Information Systems 2, pp. 171-185.

Bown, H.G., O'Brien, C.D., Sawchuk, W., et Storey, J.R. (1978). Description générale de Télidon: Proposition canadienne concernant les systèmes vidéotex. Note technique no 697-F. Centre de recherches sur les communications. Ministère des Communications, Ottawa.

Bown, H., O'Brien, C.D. Sawchuk, W., Storey, J.R., et Treurniet, W.C. (1979). "Telidon Videotex and User-related Issues," Proc. Conf. on Visible Languages.

Burger, J.F. (1977). "Data Base Semantics in the EUFID System," Proc. 2nd Berkeley Workshop on Distributed Data Management and Computer Networks, pp. 202-214.

Carlson, C.R., et Kaplan, R.S. (1976). "A Generalized Access Path Model and its Application to Relational Data Base Systems," Proc. ACM SIGMOD, pp. 143-154.

Chamberlin, D.D., et al. (1976). "SEQUEL2: A Unified Approach to Data Definition, Manipulation and Control," IBM Journal of Research and Development 20, pp. 560-575.

Codd, E.F., Arnold, R.S., Cadiou, J-M., Chang, C.L., et Roussopoulos, N. (1978). RENDEZVOUS Version 1: An Experimental English-language Query Formulation System for Casual Users of Relational Data Bases Tech. rep. RJ2144, IBM Research Lab., San Jose, Calif.

Date, C.J. (1976). "An Architecture for High-level Language Database Extensions," Proc. ACM SIGMOD, pp. 101-122.

Date, C.J. (1980). "An Introduction to the Unified Database Language," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 15-32.

Deheneffe, C. et Hennebert, H. (1976). "NUL: A Navigational User's Language for a Network Structured Data Base," Proc. ACM SIGMOD, pp. 135-142.

Denny, G.H. (1977). An Introduction to SQL, A Structured Query Language, Tech. Rep. RA93, IBM Research Lab., San Jose, Calif.

Ellis, C.A., et Nutt, G.J. (1980). "Office Information Systems and Computer Science," ACM Computing Surveys 12, pp. 27-60.

Gilb, T. et Weinberg, G.M. (1977). Humanized Input: Techniques for Reliable Keyed Input. Winthrop Publishers.

Geudj, R.A., tenHagen, P.J.W., Hopgood, F.R.A., Tucker, H.A., et Duce, D.A. (eds) (1980). Methodology of Interaction, IFIP Workshop on Methodology of Interaction, Seillac, France. North-Holland, Amsterdam.

Hammer, M., Howe, W.G., Kruskal, V.J., et Wladawsky, I (1977). "A Very High Level Programming Language for Data Processing Applications" CACM 20, pp. 832-840.

Harris, L.R. (1977) "User Oriented Data Base Query with the ROBOT Natural Language Query System," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 303-311.

Hendrix, G.G. Sacerdoti, E.D., Sagalowicz, D., et Slocum, J. (1978). "Developing a Natural Language Interface to Complex Data," ACM TODS 3, pp. 105-147.

Herot, C.F. (1980). "Spatial Management of Data," ACM TODS 5, pp. 493-513.

Hogg, J. (1981). TLA: A System for Automating Form Procedures, M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto.

Hogg, J., Nierstrasz, O.M., et Tsichritzis, D.C. (1981). "Form Procedures," in Omega Alpha, Tsichritzis, D.C. (ed.), Tech. rep. CSRG-127, Computer Systems Research Group, Univ. of Toronto.

Kerschberg, L., Ozkarahan, E.A., et Pacheco, J.E.S. (1976). "A Synthetic English Query Language for a Relational Associative Processor," Proc. 2nd Intl. Conf. on Software Engineering, pp. 505-519.

Lacroix, M., et Pirotte, A. (1977a). ILL: An English Structured Query Language for Relational Data Bases," dans Architecture and Models in Data Base Management Systems, (Nijssen, G.M., ed.), pp. 237-260, North-Holland, Amsterdam.

Lacroix, M., et Pirotte, A. (1977b). "Domain-oriented Relational Languages," Proc. Intl. Conf. Very Large Data Bases, pp. 370-378.

Lipson, W., et Lapczak, O. (1976). LSL User's Manual, Tech. note 9, Computer Systems Research Group, Univ. of Toronto.

Luo, D., et Yao, S.B. (1981). "Form Operations By Example - A Language for Office Information Processing," Proc. ACM SIGMOD.

McDonald, N., et Stonebraker, M.R. (1975). "CUPID - The Friendly Query Language," Proc. ACM Pacific 75 Regional Conf., pp. 127-131.

Mantei, M. (1981). Disorientation Behaviour in Person-Computer Interaction, Ph.D. thesis, Dept. of Communications, Univ. of Southern California (à paraître).

Martin, J. (1973). Design of Man-Computer Dialogues. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Mehlmann, M. (1981). When People Use Computers - An Approach to Developing an Interface. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Moorhead, W.G. (1976). GXRAM - A Relational Data Base Interface for Graphics, Tech. rep. RJ1735, IBM Research Lab., San Jose, Calif.

Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., et Wong, H. (1975). "TORUS - Natural Language Understanding System for Data Management," Proc. 4th Intl. Joint Conf. Artificial Intelligence, pp. 414-421.

Mylopoulos, J., Borgida, A., Cohen, P., Roussopoulos, N., Tsotsos, J., et Wong, H. (1976). "TORUS: A Step toward Bridging the Gap between Data Bases and the Casual User," Information Systems 2, pp. 49-64.

Nierstrasz, O.M. (1981). Automatic Coordination and Processing of Electronic Forms in TLA (An Intelligent Office Information System), M.Sc. thesis, Dept. of Computer Science, Univ. of Toronto.

Prenner, C.J., and Rowe, L.A (1977). A Data Base Oriented Programming Language, ERL Memo, Electronic Research Laboratory, Univ. Calif. Berkeley.

Reisner, P. (1977). "Use of Psychological Experimentation as an Aid to Development of a Query Language," IEEE Trans. on Software Engineering 3, pp. 218-229.

Reisner, P. (1981). "Human Factors Studies of Database Query Languages: A Survey and Assessment," ACM Computing Surveys 13.

Reisner, P., Boyce, R.F., et Chamberlin, D.D. (1975). "Human Factors Evaluation of Two Data Base Query Languages - SQUARE and SEQUEL," Proc. AFIPS 44, NCC, pp. 447-452.

Sagalowicz, D. (1977). "IDA: An Intelligent Data Access Program," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 293-302.

Schmidt, J.W. (1977). "Some High Level Language Constructs for Data of Type Relation," ACM TODS 2, pp. 247-261.

Sharman, G.C.H. (1977). "Update-by-dialogue: An Interactive Approach to Data Base Manipulation," Proc. ACM SIGMOD, pp. 21-29.

Shneiderman, B. (1978). "Improving the Human Factors Aspect of Database Interactions" ACM TODS 3, pp. 417-439.

Shneiderman, B. (1980). Software Psychology: Human Factors in Computer and Information Systems. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Spath, C.R. et Schneider, L.S. (1977). "A Generalized End-user Facility Architecture for Relational Data Base Systems," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 359-369.

Stonebraker, M.R., et Rowe, L.A. (1977). "Observations on Data Manipulation Languages and Their Embedding in General Purpose Programming Languages," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 128-143.

Thomas, J.C. (1977). "Psychological Issues in Data Base Management," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 169-185.

Thomas, J.C., et Gould, J.D. (1975). "A Psychological Study of Query-by-example," Proc. AFIPS 44, NCC, pp. 439-445.

Vandijck, E. (1977). "Towards a More Familiar Relational Retrieval Language," Information Systems 2, pp. 159-169

Vassiliou, Y., et Lochovsky, F.H. (1980). "DBMS Transaction Translation," Proc. COMPSAC'80, pp. 89-96.

Woods, W.A. (1973). "Progress in Natural Language Understanding - An Application to Lunar Geology," Proc. AFIPS 42, NCC, pp. 441-450.

Zloof, M.M. (1975a). "Query-by-example: The Invocation and Definition of Tables and Forms," Proc. ACM Intl. Conf. Very Large Data Bases, pp. 1-24.

Zloof, M.M. (1975b). "Query-by-example," Proc. AFIPS 44, NCC, pp. 431-433.

Zloof, M.M. (1977). "Query-by-example: A Data Base Language," IBM Systems Journal 16, pp. 324-343.

Zloof, M.M. (1980). A Language for Office and Business Automation, IBM Tech. rep. RC8091, Yorktown Heights, New York.